Department of Computing
Imperial College of Science, Technology and Medicine
University of London

Changing the Past: Database Applications of Two-Dimensional Temporal Logics

Marcelo Finger

February 1994

A thesis submited for the degree of
Doctor of Philosophy of the University of London

Abstract

Temporal active databases are an extension of the traditional database concept. On the temporal side, data is associated to time points or periods and, on the active side, temporal rules provide a means by which some temporal data in the database may force the insertion/deletion of some other temporal data. Thus, the temporal rules establish a "temporal link" between the data, either via a procedural execution mechanism or through a declarative historical interpretation. In this database framework, we are interested in analysing and detecting the effects that a change in (data associated with) the past may cause in the database.

Temporal logic techniques are applied to describe and analyse the problems related to the evolution of temporal databases. For that, we start by presenting the mathematical framework of *two-dimensional temporal logics*. Several methods of combining two one-dimensional temporal logics are described. Each combination method generates a family of two-dimensional temporal logics. The objective of this study is to establish for each combination method whether some logical properties of the one-dimensional systems, *e.g.* completeness and decidability, are transferred to their two-dimensional combination.

A temporal logical description of data representation and temporal queries in temporal databases is provided, and two-dimensional temporal logic is used to describe temporal data evolution through updates. We use this two-dimensional description to characterise the differences between the *transaction-time* and *valid-time* types of temporal databases.

A two-dimensional imperative *execution semantics* and a one-dimensional declarative *valid-time interpretation* for temporal rules are then presented. It is shown that, under the execution semantics, updates in the past, and in general any historical update, may cause a violation of the valid-time interpretation, generating a *time paradox*. A classification of time paradoxes is proposed and their occurrences are then considered as the *effects of changing the past*. Finally, several algorithms for the detection of occurrences of some types of time paradoxes are presented and their correctness is proved.

to Monika

Acknowledgements

I am deeply grateful to my supervisor, Prof. Dov M. Gabbay, who supported and encouraged me from my early days as an MSc. student until the end. The influence of his ideas, advice and guidance were of great value throughout the development of this work. I would like to thank him especially for his constant support throughout the period I was unable to work due to strain injuries on my arms.

Very special thanks to Leonardo Lazarte and Pedro Savadovski, for their almost paternal support, advice and numerous discussions on matters ranging from Logic to marriage, to life in general; and to my friend Ben Strulo, for several illuminating discussions, and for reading and commenting all my papers and an earlier draft of this thesis.

I am also thankful to my friends from the Temporal Logic Group for numerous discussions on logic, their comments on my work and for their patience in answering the unstoppable flow of my questions: Ian Hodkinson, Robin Hirsh, Peter McBrien, Richard Owens and Mark Reynolds.

Thanks to Ralph Birnbaum for disagreeing with me for three years during our lunch time. And thanks to Tony Hunter for the pleasant tea breaks and for the numerous discussion on Logic and not-so-logical topics.

Thanks to all my colleagues that helped in the development of this work and in making our life in England a wonderful experience: Paulo Azevedo, Ralph Birnbaum, Fátima Dargam, David Evans, Felipe França, João Gondim, Luis Hanna, Tony Hunter, Priscila Lima, Roberto Lins, Juarez Muylaert-Filho, Claudia Oliveira, Afonso Pinto, Francisco Simplício and Alessandra Russo.

I would also like to thank Sue Brooks and Janice Lonsdale for being always friendly and helpful.

I am thankful to the CAPES, Ministry of Education, Brazil, for their financial support, grant 1491/89.

I am thankful to those who helped me when I was suffering from repetitive strain injuries: Dr. Gabriel Panayi, the rheumatologist, Sally Joyce Waters, the physiotherapist, Garnet B. Symonds, the osteopath, and Dr. Carla Finger, my sister. I would also like to thank Kostas Stathis for typing a few pages of this thesis.

Very special thanks to my parents, Salezy and Rosa, for their constant support and their constant preoccupation about my health.

Above all, I am most grateful to my wife Monika. For the constant support, love, tenderness and lively incentive. In those disheartening moments when I could not use my hands, when I had to type with my feet or simply could not work, she

always had a smile, an infinite patience in doing things for me. I knew I could rely on her. Thank you for your company, Mo.

Contents

Chapter 1

Introduction

1.1 Motivation

The developments of database technology have extended the traditional relational database concept in several directions. Of particular interest to this work are the *temporal* extension [Snodgrass and Ahn 1985] and the *active* extension [Morgenstern 1983; Stonebraker, Hanson and Potamianos 1988].

In the temporal extension, the data is associated with a time point or period, and the query language is extended to deal with temporal queries. In such a temporal database, the traditional schema and data of a relation in the database is enhanced with a set of attributes and time related data, adding to the database a temporal dimension that it originally lacked. For example, consider the non-temporal relation `EMPLOYEE` of a traditional relational database:

NAME	SALARY	DEPARTMENT
Peter	2000	Marketing
Mary	3000	Finance

Suppose the current time is April 1993 ($Apr93$). In a temporal database, a temporal version of the above relation, with its extended schema and temporal data, could be:

NAME	SALARY	DEPT	START TIME	END TIME
Peter	1000	R&D	$Jan90$	$Dec91$
Peter	2000	Marketing	$Feb92$	$Apr93$
Mary	3000	Finance	$Sep91$	$Apr93$

In such a temporal relation, the attributes `START TIME` and `END TIME` have a special status, receiving a temporal semantical interpretation so that it is possible

to pose queries that were not directly supported by the non-temporal system. For instance, with respect to the previous (very simple) temporal database relation, one may want to know "when did Peter change departments and what was his salary increase then?" Or, one may wish to ask "who are the employees that have been working for more than two years?" Temporal query languages provide a means to pose those questions to temporal databases in a straightforward way, and the extra functionality of temporal databases, not present in traditional relational databases, allows for the correct interpretation of the special attributes so as to generate the right answers for the queries; the extra functionality needed for temporal relational databases discussed in [McBrien 1992].

The temporal data may possess distinct semantic interpretations. On the one hand, the temporal data may be interpreted as the history of the Universe of Discourse, *i.e.* the temporal evolution of the objects of the part of the real world the database is supposed to model; in this case, there is no direct relation between the time the data is entered in the database and the time in the modelled world the data refers to. For example, in the previous temporal relation, the fact that Peter started working at $Jan90$ might have been recorded at a later time, *e.g.* at $Feb90$. This semantical interpretation is called the *valid-time semantics*.

On the other hand, a different interpretation of the temporal data is possible, where the temporal attributes are seen as referring to the history of the database system, as opposed to that of the Universe of Discourse. Under such interpretation, the previous temporal relation tells us that the information that Peter has started working at the R&D department was inserted in the database at $Jan90$ and deleted at $Dec91$, independently of when, in reality, Peter has actually started or finished working at that department. This second semantical interpretation of temporal data is called the *transaction-time semantics*.

It is one of the aims of this work to provide a clear, formal distinction between those two kinds of semantical interpretations of the temporal data.

The other extension of the relational data model we wish to consider is the active database extension. This extension aims at transforming the database system from a passive repository of facts to one that is able to react to the data it contains. Such an extension is achieved by equipping the database with a set of rules. For example, the printing of the payment cheque of an employee may be specified in a rule expressed in a logic language, in the following way:

```
if    exists Dept such_that employee(Name, Salary, Dept)
then  print_payment_cheque(  Name, Salary )
```

It is also the case that there are several semantics and languages for these kinds of rules, and those semantics are either procedural or declarative in nature. Under a procedural semantics, the condition part of the rule is seen as a query that, when satisfied, fires an action (the then-part of the rule) in the database. The declarative semantics of rules is normally associated with deductive rules, so as to enable more complex queries; under such semantics, the database does not become an active system, for the then-part of the rule is seen as data deduced from the database, rather than an action to be executed. For a broad discussion on logic languages for databases and their semantics, expressivity and complexity refer to [Abiteboul and Vianu 1991].

Recently, both temporal and active extensions were simultaneously applied, generating an Active Temporal Database [Manning and Torsun 1989; Loucopoulos *et al.* 1990], where the temporal rules provide "temporal links" between data associated to different times. In such case, the previous rule concerning salary payment can be formulated as the following temporal rule, expressed in the ERL temporal language of the TEMPORA system [McBrien *et al.* 1991]:

```
if    time_is end_of_this_month and
      (employee.X has salary.S
      at start_of_this_month)
then  print_payment_cheque( X, S )
```

In such a rule, the temporal aspects are explicitly stated, and we can see that the rule will be triggered at the end of the month to pay for an employee's salary as it stood at the beginning of the month. The use of temporal active rules raises new issues on their semantics with respect to the dichotomy between procedural and declarative semantics, and those issues will be discussed in Chapter 5.

This temporal active database scenario provides a framework in which it is possible to change the data that is recorded about the past. In such a case, it makes sense to pose the following question, which is the central motivation for this thesis.

Question 1.1 *How is history affected if we change the past?*

Although the question above may constitute an adequate philosophical question deserving an adequate philosophical treatment, we are primarily concerned with the logical and computational aspects arising from such a question in the context of databases.

It is easy to imagine a situation where it is plausible to change the past recorded in the database. For example, an employee may be retroactively hired; he or she

may receive a retroactive salary increase; his or her name may have been incorrectly typed in, and later this mistake is rectified. The problem is, several actions may have been executed by the database using information that has later changed and become false. It is the goal of this work to study what are the effects generated by those changes and how to detect them.

So far we have set the framework in which to place the problem we are trying to solve, as expressed by Question 1.1. It is now necessary to set a framework in which to search for an answer. This theoretical framework is suggested by the foundations underlying the concept of a relational database. Note that classical logic is the underlying framework for the relational database; it is also the basis for the relational calculus, serving as the theoretical support to relational query languages such as SQL. The same is true for temporal logic with respect to temporal databases. Logic is also the basis for procedural/declarative interpretation of rules; actually, rule-based systems is one area where the interests of logic, artificial intelligence (in the form of expert systems) and databases converge. Temporal logic is the natural candidate to deal with temporal rules. Moreover, the notions of "change" and "evolution" (through updates), that are integral part of Question 1.1, are most naturally dealt with in temporal terms. Therefore, temporal logic is the framework chosen to investigate Question 1.1; we discuss the basic notions of temporal logic in the Background Section 1.2.

The aim of this thesis can then be set as to give a temporal logic treatment for Question 1.1. In fact, instead of being limited to changes of the past, we consider the generalisation *how is any temporal information affected when history is changed?*, which for the purposes of this thesis translates into the question of how the interpretation of temporal data stored in a database is affected when any data is changed in the presence of active "temporal links".

Question 1.1, or its generalised form, motivates several other questions which need to be answered if an answer to the original one is to be given.

Is not "change" already a temporal notion?

What is the meaning of "changing the past"?

How can the past or any time in history be changed?

Finally, what are the effects of "changing the past" and how can they be detected?

These motivated questions provide the guidelines that will lead us throughout this thesis. Consider the first of them, which is concerned with the puzzling nature of the expressions "change the past" or "change the history".

Question 1.2 *Is not "change" already a temporal notion?*

In this thesis we maintain that the answer to Question 1.2 is yes, and a temporal logic basis for that answer is given in Chapter 2. There, a means to describe the evolution of any logic system L by adding to it a temporal dimension is presented, in a process called *temporalisation*. The aim is to be able to describe the temporal evolution of a system specified in a generic logic L. The temporalisation of a generic system L with respect to a temporal logic T generates a new logic system $\mathsf{T}(\mathsf{L})$, and Chapter 2 studies how the logical properties of systems T and L, such as soundness, completeness and decidability, are transferred to the combined system $\mathsf{T}(\mathsf{L})$.

Once it is established that the notion of change is a temporal one, we already have an initial mathematical framework based on temporal logic to analyse Question 1.1. The next question calls for a broadening of such a temporal logic framework.

Question 1.3 *What is the meaning of "changing the past"? Or, in general, what is the meaning of changing any time in history at all?*

To answer that question, the explicit double temporality of "change" on the one hand, and of "past" or "history" on the other, as in the expressions "changing the past" and "changing the history", is investigated under a formal logic point of view. Logics for two-dimensional time are developed, in which one temporal dimension contains a description of the history of modelled reality, corresponding to the previously mentioned valid-time interpretation of temporal data, while the other dimension describes the evolution of how the history of modelled reality is seen at different times, corresponding to the transaction-time interpretation of temporal data.

Two-dimensional logics over the two-dimensional plane often do not posses the desired properties. As it is shown in Section 3.3, although there are complete axiomatisations of one-dimensional temporal logics over several linear classes of flows of time, sometimes it is impossible to obtain complete axiomatisations over the two-dimensional plane, *e.g.* over $\mathbb{Z}\times\mathbb{Z}$ and $\mathbb{R}\times\mathbb{R}$. Therefore, we study other possible two-dimensional temporal logics that are weaker than the full two-dimensional case but that succeed in transferring all several logical properties from the one-dimensional case to the two-dimensional one.

A great number of temporal logics exist in the literature to deal with the great variety of properties one may wish to assign to flows of time. In building two-dimensional temporal logics, the combination of two classes of flows of time generates an even greater number of possible systems to be studied. It is, therefore, desirable to study if it is possible to transfer the properties of long known and studied (one-dimensional) temporal logic systems to the two-dimensional case.

One possible way to obtain such a two-dimensional temporal logic is to apply the temporalisation process to a temporal logic, generating the system $\mathsf{T}_1(\mathsf{T}_2)$. However, as we shall see, the logic system $\mathsf{T}_1(\mathsf{T}_2)$ is very limited in its expressivity, so we have to look for stronger systems. So in Chapter 3 we propose other methods for combining two one-dimensional temporal logics so as to obtain more expressive two-dimensional systems. As in Chapter 2, the emphasis continues to be on studying how the logical properties transfer from the component logic systems T_1 and T_2 to the combined system according to each combination method.

Changes in history are then seen as a two-dimensional temporal evolution. To answer the next motivated question it is necessary to move from the abstract pure logic framework to a more data oriented one.

Question 1.4 *How can the past or any time in history be changed?*

Question 1.4 calls for a representation of temporal data, so that we read "changing the history" as "changing the recorded history". Chapter 4 hence defines the notions of temporal database, temporal data representation and temporal queries in terms of temporal logic. A two-dimensional update semantics is provided, in terms of which a formal distinction of the two distinct notions of temporal databases, namely transaction-time databases and valid-time databases, is presented.

The mere fact that temporal data can be updated does not imply that updating data at one time will have any effect on some other data. In Chapter 5 the valid-time database is therefore enhanced with temporal rules, so as to provide "temporal links" between the data in the database. In this context, the final motivated question can be explored.

Question 1.5 *What are the effects of changing the past? More generally, what are the effects of changing data associated to any particular time? And how can these effects be detected?*

The temporal rules in the active valid-time database are equipped with an imperative two-dimensional semantics and with a declarative one-dimensional valid-time

interpretation. The imperative interpretation views rules as applicable only at the current execution time, but the declarative interpretation looks at rules as valid-time constraints holding at all times. For instance, over discrete time, the rule

```
if    Condition
then  Action
```

can be imperatively interpreted by, at the current time, checking whether `Condition` holds and in case of success execute `Action`, and this process is repeated whenever time is advanced, changing the value of the current time. Alternatively, this rule can be seen as a historical law, such that at every (valid-) time point, and not only at the current one, whenever `Condition` holds `Action` (that is seen as a formula that can be checked against the database state just as `Condition` can) must hold.

It is shown that, due to the occurrence of updates in the past or, in general, at any time, the execution semantics may cause the valid-time interpretation of the rules to become invalid. The occurrences of such invalidations are called *time paradoxes* and are interpreted as "the effects" of changing the past or history itself. The next step is to analyse how these time paradoxes can be algorithmically detected, which we accomplish in the following way. A classification of time paradoxes is proposed based on the different interactions between updates, rule execution and the violation of the valid-time interpretation. Finally, several algorithms for the detection of several types of time paradoxes are presented and their correctness is proved.

1.1.1 The Organisation of the Thesis

This thesis clearly has two distinct parts, one being formal logic oriented while the other is database oriented. In the first part, several methods of combination of temporal logics are studied. The aim is to determine if the logical properties of the combined system are transferred through each method. As a result, several two-dimensional temporal logics are described, with varying degrees of expressivity. The second part is concerned with temporal databases and their evolution through updates. Two-dimensional temporal logics are applied in the description of this evolution. The aim is to determine what are the effects of updating history and how to detect them.

Throughout this thesis temporal logic—and logic in general—plays a central role. Therefore those notions are described in the Background Section 1.2. Other basic notions are introduced in the body of the thesis as they are needed, mainly in the initial section of each chapter. For instance, the formal definition of a temporal

database is only introduced in Chapter 4; and the presentation of temporal rules is delayed until temporal active databases are discussed in Chapter 5.

With respect to its degree of generality and abstraction, this thesis moves from a very abstract start to a more practically oriented ending. Chapter 2 presents a very general combination of logics in which a temporal logic is combined to a generic logic system via the temporalisation process. Chapter 3 concentrates on methods for the combination of two temporal logics, generating several two-dimensional logics. In Chapter 4, the focus is moved from logic to databases, and two-dimensional temporal logics are applied in the description of updates in temporal databases. Chapter 5 enhances the database with temporal rules and discusses the effects of updates in temporal active databases; several algorithms are proposed for the detection of such effects. Finally, Chapter 6 discusses the results of the thesis, compares them with the literature and suggests further areas of research. Appendix A collects the algorithms developed in the thesis and Appendix B contains the proofs of some auxiliary results that were used or cited in the body of the thesis.

1.1.2 Published Material

The contents of Chapter 2 have appeared in [Finger and Gabbay 1992a] and will soon appear in [Gabbay, Hodkinson and Reynolds 1994, Chapter 14]. Chapters 4 and 5 reorganise and largely extend the material that appeared published in [Finger 1992] and [Finger and Gabbay 1992b]. All those papers where developed by the author working under the supervision of Prof. D. M. Gabbay.

The other papers in which this author appears cited as a coauthor were produced as part of ESPRIT project TEMPORA and do not play a central role in this thesis [Finger, McBrien and Owens 1991; Finger, Fisher and Owens 1993].

1.1.3 Statement of Contribution

The contributions of this thesis are the following.

In Chapter 2, the temporalisation process and its property transference results are all original.

In Chapter 3, the results about independent combination of logics extend the results of [Kracht and Wolter 1991; Fine and Schurz 1991] to the case of the non-independent temporal modalities; Theorem 3.2 extends a result of [Venema 1990] for the two-dimensional plane and for distinct flows of time; the results about restricted interlacing are all original and answer a conjecture of Venema [1990].

The two-dimensional characterisation of transaction-time databases is the contribution of Chapter 4.

Finally, the treatment of temporal dependences and the algorithms to detect time paradoxes originated by conflicts between imperative and declarative semantics of temporal rules are contributions of Chapter 5.

1.2 Background

This section presents the basic definitions that will be used constantly throughout the thesis. The main concern here is to define logic systems in general, and temporal logics in particular, while in the next two chapters the discussion will focus on the combination of several temporal logic systems.

1.2.1 Formal Logic Systems

The modern notion of Logic dates back to Frege [1879] and the influential works of Whitehead and Russell [1910] and Hilbert [1925; 1927]. Currently, there are several approaches to define what a logic system is, namely the syntactical approach, the semantical approach and the algebraic approach. This work concentrates in the first two ones.

The syntactical approach is concerned with the inferences that can be obtained from a given, possibly empty set of premises. The entities that are relevant to this approach belong to the pair $\langle \mathcal{L}_{\mathsf{L}}, \vdash_{\mathsf{L}} \rangle$, where $\mathcal{L}_{\mathsf{L}}$ is its *language* and $\vdash_{\mathsf{L}}$ is its *inference system*; the language $\mathcal{L}_{\mathsf{L}}$ is a set of well-formed sequences of symbols called *formulae*, constructed from an alphabet Σ_{L} of symbols according to a set of *formation rules*; the inference system $\vdash_{\mathsf{L}}$ is a relation between sets of formulae and formulae of $\mathcal{L}_{\mathsf{L}}$, *i.e.* a relation between premises and conclusions, and if $A \in \mathcal{L}_{\mathsf{L}}$ then, when the set of premises is empty, $\vdash_{\mathsf{L}} A$ stands for $\varnothing \vdash_{\mathsf{L}} A$.

The semantical approach is concerned with the truth of statements. Truth is evaluated with respect to mathematical structures called *models*; a model for the logic L is a structure $\mathcal{M}_{\mathsf{L}}$ and we denote $\mathcal{M}_{\mathsf{L}} \models A$ when a formula $A \in \mathcal{L}_{\mathsf{L}}$ is true under the model $\mathcal{M}_{\mathsf{L}}$; the class of all models of L is denoted by $\mathcal{K}_{\mathsf{L}}$. The expression "class of models" should be substituted by "set of models", but it is so deeply buried in the tradition of mathematical logic that we have to keep it.

For the purposes of combination of logics which will be considered later, a *logic system* L contains syntactical and semantical elements and consists of a language, an inference system and a class of model structures.

Example 1.1 Consider the logic system PC of propositional classical logic. Its vocabulary Σ_{PC} consists of the boolean connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$, the punctuation symbols '(' and ')', and a countable set of propositional letters $\mathcal{P}$. The formation rules for $\mathcal{L}_{\mathsf{PC}}$ are:

- $\mathcal{P} \subset \mathcal{L}_{\mathsf{PC}}$;
- if $A \in \mathcal{L}_{\mathsf{PC}}$ then $\neg A \in \mathcal{L}_{\mathsf{PC}}$;
- if $A, B \in \mathcal{L}_{\mathsf{PC}}$ then $(A \wedge B), (A \vee B), (A \rightarrow B) \in \mathcal{L}_{\mathsf{PC}}$; (the parentheses are omitted when no ambiguity is implied and the precedence order $\neg$, $\wedge$, $\vee$, $\rightarrow$ is respected)
- nothing else is in $\mathcal{L}_{\mathsf{PC}}$.

The inference relation, $\vdash_{\mathsf{PC}}$, will be presented here as an axiomatisation, consisting of axioms and inference rules. The axioms are

(a) $p \rightarrow (q \rightarrow p)$
(b) $(p \rightarrow (p \rightarrow q)) \rightarrow (p \rightarrow q)$
(c) $(p \rightarrow (q \rightarrow r)) \rightarrow (q \rightarrow (p \rightarrow r))$
(d) $(q \rightarrow r) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$
(e) $p \wedge q \rightarrow p$
(f) $p \wedge q \rightarrow q$
(g) $p \rightarrow (q \rightarrow p \wedge q)$
(h) $p \rightarrow p \vee q$
(i) $q \rightarrow p \vee q$
(j) $((p \rightarrow r) \wedge (q \rightarrow r)) \rightarrow ((p \vee q) \rightarrow r)$
(k) $(p \rightarrow q \wedge \neg q) \rightarrow \neg p$
(l) $\neg\neg p \rightarrow p$

The rules of inference are Modus Ponens: from A and $A \rightarrow B$ infer B; and Substitution: from an axiom $A(q)$ infer $A(q \backslash B)$, where the latter is the formula obtained by substituting all the occurrence of the the propositional letter q by the formula B in A. If Δ is a set of formulae and A is a formula, we write $\Delta \vdash_{\mathsf{PC}} A$ if there exists a finite sequence of formulae ending in A such that each formula in the sequence is either an axiom, or belongs to Δ, or is obtained from previous formulae in the sequence through the use of an inference rule.

Finally, to define a model $\mathcal{M}_{\mathsf{PC}}$ for propositional classical logic, let $TV = \{true, false\}$ be a set of truth values and consider a valuation function $v : \mathcal{P} \rightarrow TV$, such that every propositional letter is associated to a truth value. The model $\mathcal{M}_{\mathsf{PC}}$ is a structure composed of $\mathcal{M}_{\mathsf{PC}} = (TV, v)$, so that the class of all models contains all the possible valuation functions. A formula of $\mathcal{L}_{\mathsf{PC}}$ that is true in all models The truth of a formula A in a model $\mathcal{M}_{\mathsf{PC}}$, represented by $\mathcal{M}_{\mathsf{PC}} \models A$, is given by:

$\mathcal{M} \models p$ iff $p \in \mathcal{P}$ and $v(p) = true$.
$\mathcal{M} \models \neg A$ iff it is not the case that $\mathcal{M} \models A$.
$\mathcal{M} \models A \wedge B$ iff $\mathcal{M} \models A$ and $\mathcal{M} \models B$.
$\mathcal{M} \models A \vee B$ iff $\mathcal{M} \models A$ or $\mathcal{M} \models B$.
$\mathcal{M} \models A \rightarrow B$ iff $\mathcal{M} \not\models A$ or $\mathcal{M} \models B$. □

Let A be any formula in $\mathcal{L}_{\mathsf{L}}$. A logic system L is said to be *sound* if, whenever $\vdash_{\mathsf{L}} A$, we have $\mathcal{M}_{\mathsf{L}} \models A$ for all $\mathcal{M}_{\mathsf{L}} \in \mathcal{K}_{\mathsf{L}}$. The logic system L is said to be *complete* if, whenever $\mathcal{M}_{\mathsf{L}} \models A$ for all $\mathcal{M}_{\mathsf{L}} \in \mathcal{K}_{\mathsf{L}}$, we have that $\vdash_{\mathsf{L}} A$. A formula A is *valid* in L if $\mathcal{M}_{\mathsf{L}} \models A$ for all $\mathcal{M}_{\mathsf{L}} \in \mathcal{K}_{\mathsf{L}}$ and the *validity problem* for L consists of determining whether a given $A \in \mathcal{L}_{\mathsf{L}}$ is valid or not. A formula A is a *theorem* of L if $\vdash_{\mathsf{L}} A$ and the *decision problem* for L consists of determining whether a given $A \in \mathcal{L}_{\mathsf{L}}$ is a theorem or not. Soundness, therefore, holds for L if being a theorem implies being a valid formula; completeness holds if being a valid formula implies being a theorem. For example, the propositional classical logic PC is sound and complete, so all propositinal tautologies, *i.e.* valid formulae, are theorems and vice-versa.

1.2.2 Propositional Temporal Logics

This presentation of Temporal Logic is the result of enhancing the vocabulary of classical propositional logic with modal operators. The operators approach to temporal logics started with Prior [1957], with the one-place operators P and F, which were then called *tense operators*, while the logic they generated was called *tense logic*; an alternative approach to temporal reasoning handles temporal features in a first-order language, *e.g.* [Kowalski and Sergot 1986]. In Prior's approach, if A is a proposition, the formula of the form PA reads as "sometime in the past, it was the case that A" and similarly FA reads as "sometime in the future, it will be the case that A". The purpose of the tense operators was to capture tenses in natural language sentences; since then, the operators approach has been applied

to describe a variety of other "non-tense" temporal systems, *e.g.* in artificial intelligence [Halpern and Shoham 1986], in software engineering [Pnueli 1977; Kröger 1987] and in databases [Tuzhilin and Clifford 1990; Gabbay and McBrien 1991]. The name *temporal logics* has been appropriated to encompass this wide area of applications.

The Prior F and P operators are not as expressive as the two-place temporal operators "Since" (S) and "Until" (U) introduced by Kamp [1968]. A formula of the form $S(A, B)$ is read as "since A was the case, B has been the case" and similarly "$U(A, B)$" is read as "until A is the case, B will be the case."

We present here several propositional temporal logics of "Since" and "Until"; these logics are defined over the same language but vary in the nature of the flow of time they describe. So the language is defined starting from a countable set of propositional letters $\mathcal{P}$ and then formulas are built up from the propositional letters using the boolean operators $\neg$ (negation) and $\wedge$ (conjunction) and the two-place temporal operators S (since) and U (until). Other boolean connectives such as $\vee$ (disjunction), $\rightarrow$ (material implication) and $\leftrightarrow$ (material biconditional), as well as the abbreviations $\top$ (constant true) and $\bot$ (constant false), can be defined in terms of $\neg$ and $\wedge$ in a standard way; similarly for other temporal operators like P (sometime in the past), F (sometime in the future), H (always in the past) and G (always in the future) with respect to S and U.

In the following, propositional letters are represented by p, q, r and s, and temporal formulae are represented by upper case letter A, B, C and D.

Definition 1.1 Syntax of propositional temporal logics Let $\mathcal{P}$ be a countably infinite set of propositional letters. The set $\mathcal{L}_{\mathsf{US}}$ of temporal propositional formulas is the smallest set such that:

- $\mathcal{P} \subset \mathcal{L}_{\mathsf{US}}$;
- If A and B are in $\mathcal{L}_{\mathsf{US}}$, then $\neg A$ and $(A \wedge B)$ are in $\mathcal{L}_{\mathsf{US}}$;
- If A and B are in $\mathcal{L}_{\mathsf{US}}$, then $S(A, B)$ and $U(A, B)$ are in $\mathcal{L}_{\mathsf{US}}$.

The *mirror image* of a formula is another formula obtained by swapping all occurrences of U by S and vice-versa. □

The brackets of a formula are sometimes omitted when no ambiguity is implied. Boolean connectives are defined in the standard way, while temporal operators can be defined by:

$$
\begin{aligned}
FA &=_{def} U(A, \top) \\
PA &=_{def} S(A, \top) \\
GA &=_{def} \neg F \neg A \\
HA &=_{def} \neg P \neg A
\end{aligned}
$$

The ontology of time has to be defined before we are able to provide a semantics to temporal formulae. According to van Bentehm [1983] there are three basic ontologies of time to be considered, namely:

(a) points;

(b) intervals;

(c) events.

In this work we concentrate mainly on a point-based temporal ontology. The interval based cases have been discussed by [Halpern and Shoham 1986] and [Venema 1990] (the latter has shown a relationship between interval based temporal logics and two-dimensional temporal logics that is of interest to our work here). The event-based approach to temporal reasoning has been investigated on the lines of the Event Calculus of [Kowalski and Sergot 1986].

Under the point-based ontology, a flow of time is an ordered pair $\mathcal{F} = (T, <)$, where T is a possibly infinite, nonempty set of time points and $<$ is a binary relation over T. Several restrictions can be made to the nature of the flow of time; the following properties are among the most frequent ones encountered in the literature; for a first- and second-order formulation of several other properties, refer to [Burgess 1984].

(a) *irreflexivity:* for no $t \in T$, $t < t$;

(b) *transitivity:* for all $s, t, u \in T$, if $s < t$ and $t < u$ then $s < u$;

(c) *totality:* for all $s, t \in T$, either $s < t$ or $s = t$ or $t < s$.

(d) *linearity:* irreflexivity, transitivity and totality.

(e) *boundedness:* there exist $t_{min}, t_{max} \in T$ such that all $t \in T$, $t_{min} \leq t \leq t_{max}$ ($a \leq b$ is the usual abreviation for $a < b$ or $a = b$);

(f) *unboundedness:* for every $t \in T$, there exist $s, u \in T$ such that $u < t < s$;

(g) *discreteness:* for every $t \in T$, if there exists $s \in T$, $t < s$ then there exist a $t' \in T$, the successor of t, such that $t < t'$ and for no $u \in T$, $t < u < t'$; and if there exists $s \in T$, $s < t$ then there exist a $t'' \in T$, the predecessor of t, such that $t'' < t$ and for no $u \in T$, $t'' < u < t$.

(h) *denseness:* for every $s, t \in T$ such that $s < t$, there exists $u \in T$, $s < u < t$;

(i) $\mathbb{Z}$*-like:* linearity, discreteness, unboundedness such that between every two points of T there are only finitely many points;

(j) $\mathbb{Q}$*-like:* linearity, denseness, unboundedness such that T is countable;

(k) $\mathbb{R}$*-like:* linearity, denseness, unboundedness such that T contains all least upper bounds and greatest lower bounds of sequences of elements of T.

The first two properties of irreflexivity and transitivity are sometimes imposed to all flows of times, but we follow [Burgess 1984] by initially treating $<$ as a generic relation on the set of time points T. Later in the thesis the $<$ relation will be actually constrained to a linear order, therefore encompassing the properties of irreflexivity and transitivity.

Definition 1.2 Semantics of propositional temporal logic

A valuation g is a function assigning to every time point t in T a set of propositional letters $g(t) \subseteq \mathcal{P}$, namely the set of proposition letters that are true at the time point t.[1] A *model* $\mathcal{M}$ is a 3-tuple $(T, <, g)$, where $(T, <)$ is the underlying flow of time and g is a valuation. $\mathcal{M}, t \models A$ reads "the formula A holds over model $\mathcal{M}$ at time point t" and is defined recursively as follows.

$\mathcal{M}, t \models p$ iff $p \in \mathcal{P}$ such that $p \in g(t)$.

$\mathcal{M}, t \models \neg A$ iff it is not the case that $\mathcal{M}, t \models A$.

$\mathcal{M}, t \models A \wedge B$ iff $\mathcal{M}, t \models A$ and $\mathcal{M}, t \models B$.

$\mathcal{M}, t \models S(A, B)$ iff there exists an $s \in T$ with $s < t$ and $\mathcal{M}, s \models A$ and for every $u \in T$, if $s < u < t$ then $\mathcal{M}, u \models B$.

$\mathcal{M}, t \models U(A, B)$ iff there exists an $s \in T$ with $t < s$ and $\mathcal{M}, s \models A$ and for every $u \in T$, if $t < u < s$ then $\mathcal{M}, u \models B$.

□

A formula A is *valid* over a class $\mathcal{K}$ of flows of time, indicated by $\mathcal{K} \models A$, if for every $\mathcal{M}$ whose underlying flow of time is in $\mathcal{K}$ and for every time point $t \in T$, $\mathcal{M}, t \models A$. If Σ is a set of formulae, we write $\mathcal{K} \models \Sigma$ to indicate that $\mathcal{K} \models A$ for every $A \in \Sigma$. Therefore, for different classes $\mathcal{K}$ we have different sets of valid formulae.

The inference system of temporal logics is given here in the form of Hilbert Axiom Systems [Hilbert 1925; 1927]. Such systems are composed of a set of *axioms*

[1] Alternatively a valuation could be defined as a function $h : \mathcal{P} \to 2^T$, associating every propositional letter to a set of time points in which it holds true [Burgess 1984; Gabbay, Hodkinson and Reynolds 1994].

and a set of *inference rules*. An axiomatic system for the US-temporal logic over the class of all flows of time $\mathcal{K}_0$, $\vdash_{\mathsf{US}}$, contains the following axioms:

A0 all classical tautologies

A1a $G(p \to q) \to (U(p,r) \to U(q,r))$

A1b $H(p \to q) \to (S(p,r) \to S(q,r))$

A2a $G(p \to q) \to (U(r,p) \to U(r,q))$

A2b $H(p \to q) \to (S(r,p) \to S(r,q))$

A3a $(p \wedge U(q,r)) \to U(q \wedge S(p,r),r)$

A3b $(p \wedge S(q,r)) \to S(q \wedge U(p,r),r)$

Note that the axioms above come in pairs, represented by **a** and **b**, such that one is the mirror image of the other. The inference rules are:

Subst Uniform Substitution, i.e. let $A(q)$ be an axiom containing the propositional letter q and let B be any formula, then from $\vdash A(q)$ infer $\vdash A(q\backslash B)$ by substituting all appearances of q in A by B.

MP Modus ponens: from $\vdash A$ and $\vdash A \to B$ infer $\vdash B$.

TG Temporal Generalisation: from $\vdash A$ infer $\vdash HA$ and $\vdash GA$.

A *deduction* is a finite sequence of formulae each of which is either an axiom or follows from earlier formulae by a rule of inference. A *theorem* is any formula A appearing as a last element of a deduction, and we indicate this by $\vdash_{\mathsf{US}} A$. The axioms of $\vdash_{\mathsf{US}}$ can be extended by a set of axioms Σ so as to impose restrictions on the flow of time, therefore generating the inference system $\vdash_{\mathsf{US}(\Sigma)}$. When Σ is the empty set we have $\vdash_{\mathsf{US}} = \vdash_{\mathsf{US}(\varnothing)}$. A formula or set of formulae is *consistent* with respect to an inference system $\vdash$ if falsity ($\bot$) cannot be deduced from it. We abuse notation and write $A \vdash \bot$ instead of $\{A\} \vdash \bot$.

We say that an inference system, $\vdash$, is *sound* and *complete* with respect to a class $\mathcal{K}$ of flows of time if

$$\mathcal{K} \models A \text{ iff } \vdash A,$$

or equivalently,

$$A \text{ is consistent iff } A \text{ has a model over } \mathcal{K},$$

soundness corresponding to the *if* part and completeness [2] to the *only if* part. We write $\mathsf{US}/\mathcal{K}$ to indicate that US is sound and complete over the class $\mathcal{K}$ of flows of time.

Let $\mathcal{K}_0$ be the class of all flows of time, *i.e.* the set of all pairs $(T, <)$ with no special constraint imposed on $<$. Then we have the following well known result.

Theorem 1.1 (Soundness and Completeness of $\mathsf{US}/\mathcal{K}_0$)
The inference system $\vdash_{\mathsf{US}}$ is sound and complete with respect to the class $\mathcal{K}_0$.

An elegant proof of the above is given by Xu [1988]. A proof of completeness for the class of transitive linear flows of time, $\mathcal{K}_{lin}$, is given by Burgess [1982] adding the following set Σ_{lin} of axioms together with their mirror images (**b** axioms).

A4a $U(p,q) \rightarrow U(p, q \wedge U(p,q))$

A5a $U(q \wedge U(p,q), q) \rightarrow U(p,q)$

A6a $(U(p,q) \wedge U(r,s)) \rightarrow$

$$(U(p \wedge r, q \wedge s) \vee U(p \wedge s, q \wedge s) \vee U(q \wedge r, q \wedge s))$$

Burgess actually used an extra axiom, but Xu [1988] proved the same result omitting it and axiom **A5b**. Axioms **A4a**, **A4b** and **A5a** are responsible for restricting the class of flows of time to a transitive one. The pair of axioms **A6a** and **A6b** is responsible for restricting the class of flows of time to a linear one. The axiom

A7a $F\top \rightarrow U(\top, \bot)$

and its mirror image **A7b** are responsible for restricting the flow of time to a discrete one. Extending original proofs of completeness to include new axioms over a more restricted flow of time is discussed by Burgess [1984]. With axioms $\Sigma_{dis} = \Sigma_{lin} \cup \{\mathbf{A7a}, \mathbf{A7b}\}$, we have soundness and completeness results for a class of linear, discrete and transitive flows of time.

Adding the following axiom and its mirror image to Σ_{lin}, thus obtaining Σ_{dense},

A8a $\neg U(\top, \bot)$

[2] This is sometimes called *weak completeness*; strong completeness says that for any (possibly infinite) set of formula Γ, if Γ is consistent then Γ has a model. Strong completeness implies weak completeness but the converse is not true.

a complete axiomatisation over the class of all linear dense flows of time, $\mathcal{K}_{dense}$, is obtained. Curiously, that axiomatisation is also complete over the set of rational numbers $\mathbb{Q}$, implying that there is no axiom that constrains the flow of time to a countable one. There are also complete axiomatisations $\mathsf{US}/\mathbb{R}$ over the reals [Gabbay and Hodkinson 1990; Reynolds 1992] and $\mathsf{US}/\mathbb{Z}$ over the integers [Reynolds 1992].

The issue of obtaining complete axiomatisations has been a great concern in the literature of logic, in general, and temporal logic, in particular. One possible explanation for this fact[3] comes from the "boldness" of the completeness property, equating an existential property ('there exists a deduction for A') with a universally quantified sentence ('A holds in every model'). It also gives us the first indication of computability properties associated with the logic, for finite axiomatisability implies that valid formulae are recursively enumerable and that the logic is at least semi-decidable. Note, however, that axiomatisability is not the only interesting property and throughout this presentation several other properties of logic systems will be studied.

[3]as expressed by Yuri Gurevich on a lecture at Imperial College.

Chapter 2

Adding a Temporal Dimension to a Logic System

We introduce a methodology whereby an arbitrary logic system L can be enriched with temporal features to create a new system $\mathsf{T}(\mathsf{L})$. The new system is constructed by combining L with a pure propositional temporal logic T (such as linear temporal logic with "Since" and "Until") in a special way. We refer to this method as "adding a temporal dimension to L" or just "temporalising L".

We show that the logic system $\mathsf{T}(\mathsf{L})$ preserves several properties of the original temporal logic like soundness, completeness, decidability, conservativeness and separation over linear flows of time.

The temporalisation of first-order logic is presented as an example and described in detail. A comparison is then made between the modal/temporal operators approach to combining logics and other first-order approaches to the handling of time.

The temporalisation process is the first among several methods for combining temporal logic systems that will be analysed in Chapter 3.

The contents of this chapter have appeared in [Finger and Gabbay 1992a].

2.1 Introduction

We are interested in describing the way that a system $\mathcal{S}$, specified in a logic L, changes over time. There are two main methods for doing so. In the *external* method, snapshots of $\mathcal{S}$ are taken at different moments of time and each describes the state of $\mathcal{S}$ at that time. We can write $\mathcal{S}_t$ for the way $\mathcal{S}$ is at time t, and use L to describe $\mathcal{S}_t$. We then externally add a temporal system that allows us to relate different $\mathcal{S}_t$ at different times t.

In the *internal* method, instead of considering $\mathcal{S}$ as a whole, we observe how $\mathcal{S}$ is built up from internal components and we transform these components into time dependent building blocks. The internal temporal description of each component will give us the temporal description of the whole system $\mathcal{S}$. We can assume that $\mathcal{S}$ can be completely described through its components and that the way the components are put together to make $\mathcal{S}$ into a whole is also a (possibly time varying) component.

Both the external and the internal methods have their counterpart in standard temporal logic. A temporal logical system with temporal connectives such as "Since" and "Until" is the result of externally turning classical logic into a temporal (time varying) system. The use of a two-sorted predicate logic with one time variable in which atoms are of the form $A(t, x)$, with t denoting time and x denoting an element of a domain, is an internal way of making classical logic into a temporal system.

The purpose of this chapter is to investigate the external way of temporalising a logic system. In the external approach, we do not need to have detailed knowledge about the components of the system $\mathcal{S}$ or about the logical components of its description in L. We introduce a methodology whereby an arbitrary logic system L can be enriched with temporal features to create a new system $\mathsf{T}(\mathsf{L})$. The new system is constructed by combining L with a pure propositional temporal logic T (e.g. linear-time temporal logic with "Since" and "Until") in a special way. We refer to this method as "adding a temporal dimension to L" or just "temporalising L". The method we use is not confined to temporal features only, but is a methodology of combining two logics by substituting one in another. Thus in the general case we can combine any two logic systems L_1 and L_2 to form $\mathsf{L}_1(\mathsf{L}_2)$.

In classical propositional temporal logic we add to the language of classical propositional logic the connectives P and F and we are able to express statements like

in the future A will hold,

by constructing sentences of the form FA, where A is any proposition The idea we develop here is to apply temporal operators not only to propositions but also to sentences from an arbitrary logic system L.

Our aim can be viewed as describing both the "statics" and the "dynamics" of a logic system, while still remaining in a logical framework. The "statics" is given by the properties of the underlying logic system L; in propositional temporal logic T, we already have the ability to describe the "dynamics", i.e. changes in time of a set of atomic propositions. This point of view leads us to combine the upper-level temporal T system with an underlying logic system L so as to describe the evolution in time of a theory in L and its models.

Another more general point of view comes from the work in [Gabbay 1991c] about networks of logic databases. A database is considered to be a model of a theory in some logic system L_2 and the interaction between databases is modelled by another logic system L_1; therefore, two basic logic levels can be identified, namely the local logic L_2 and the global logic L_1. The two systems are illustrated in Figure 2.1 with a temporal upper-level system T in the place of L_1 and an arbitrary underlying logic system L in the place of L_2.

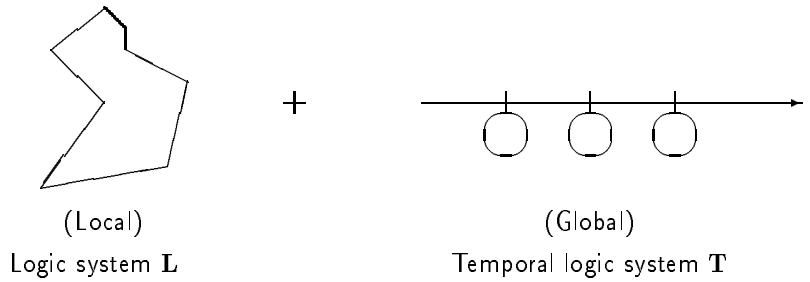


Figure 2.1 Two logic levels in a database network

We consider a network of databases distributed in time, as an extension of the more usual idea of a network of databases distributed in space. The underlying logic system L characterises the local behaviour of a database, i.e. the way queries are answered by a single element of the network. The upper-level logic system describes how one local system (at some moment in time) relates to another local system (at some other moment in time). We combine those two logic systems to be able to reason about the "temporal network" as a whole, creating a logic system $\mathsf{T}(\mathsf{L})$. The result of this combination is the addition of a temporal dimension to system L, as illustrated in Figure 2.2.

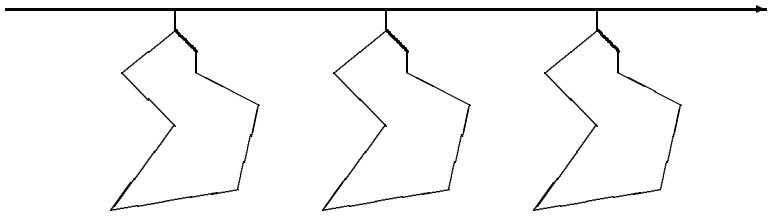


Figure 2.2 The logic system $\mathsf{T}(\mathsf{L})$

The above point of view is not yet the most general setting for our operations. One may ask a general question: given two logics L_1 and L_2, can we combine them into one logic? Suppose we take a disjoint union of the two systems, for example a modal logic system K, with modality $\Box_1$, and a modal logic system $\mathsf{S4}$, with modality

$\Box_2$. Here $\mathsf{L}_1 = \mathsf{K}$ and $\mathsf{L}_2 = \mathsf{S4}$. Form a language with $\{\Box_1, \Box_2\}$ and the separate axioms on $\Box_1$ (K axioms) and on $\Box_2$ ($\mathsf{S4}$ axioms). What do we know about the union? What is the semantics? These questions have been recently investigated by Fine and Schurz [1991] and by Kracht and Wolter [1991], in a framework in which several independently axiomatisable monomodal systems were syntactically combined. This presentation differs from the above papers in three aspects. Firstly, we are dealing with binary connectives *Since* (S) and *Until* (U). Secondly, temporal logic is a bimodal system where the two modalities, one for the past and one for the future, always interact. Thirdly, we are not arbitrarily combining two logics but rather embedding one logic inside the other. Embedding one modality within another in the framework above would syntactically combine them ruling out formulae containing $\Box_1$ within the scope of $\Box_2$. This yields what we call $\mathsf{L}_1(\mathsf{L}_2)$ (L_1 *externally* applied to L_2). The special case where L_1 is a temporal logic T and L_2 is an arbitrary logic L, gives us $\mathsf{T}(\mathsf{L})$, that we study here. We present the temporal analogue of the independent combination in Section 3.2.

General combinations of logics have been addressed in the literature in various forms. Combinations of tense and modality were discussed in [Thomason 1984], without explicitly providing a general methodology for doing so. A methodology for constructing logics of belief based on existing deductive systems was proposed by Konolige [1986]; in this case, the language of the original system was the base for the construction of a new modal language, and the modal logic system thus generated had its semantics defined in terms of the inferences of the original system. The model theory used by Konolige, called a *deductive model*, was the connection between the original system and the modal one. Here we present a quite different methodology, in which the language, inference system and semantics of $\mathsf{T}(\mathsf{L})$ are based on, respectively, the language, the inference system and the semantics of T and L.

Extensions of temporal logic are also found in the literature. In [Casanova and Furtado 1982] a family of formal languages was generated by means of certain mechanisms to define temporal modalities; the approach there was based on grammars and the resulting family of languages was claimed to be useful in expressing transition constraints for databases. Gabbay [1991b] mixes two predicate languages G and L, generating the language $\mathsf{L}^*_k(\mathsf{G})$, a two-sorted predicate language in which one sort comes from terms originated in G and the other sort comes from terms originated in L; in the case that the original language G is supposed to describe an order relation $<$, the resulting system $\mathsf{L}^*_k(\mathsf{G})$ can be seen as a predicate logic like approach

to temporal logic. Such a construction corresponds to an *internal* way of adding a temporal dimension to a logic system. We propose in this work a different approach, in which temporal modalities are applied to an existing logic system and thence a temporal dimension is added. We will informally compare the internal and external approaches in Section 2.7.

The rest of the chapter is organised as follows. In Section 2.2 we formalise the idea of temporalising a logic system L in terms of the US-temporal logic and we show the soundness and completeness of the resulting system $\mathsf{T}(\mathsf{L})$ over linear time. Section 2.3 shows that $\mathsf{T}(\mathsf{L})$ preserves the decidability property of system L over linear time, and the complexity of the decision procedure is estimated. Section 2.4 shows that $\mathsf{T}(\mathsf{L})$ is a conservative extension of L. Section 2.5 shows that $\mathsf{T}(\mathsf{L})$ has the separation property, which is useful to specify how the past states of a database influence its future states. In Section 2.6 we discuss the temporalisation of first-order logic as a particularly interesting application; two different temporalisations of first-order logic are shown, yielding two expressively different logics. Finally, in Section 2.7 we show how the added temporal dimension can be internalised in first-order logic and we compare the temporalised approach with the internalised first-order one.

2.2 Temporalising an Existing Logic

This section will construct $\mathsf{T}(\mathsf{L})$ out of T and L. Our T is the temporal system with "Since" and "Until". Our L is in general any logic and in particular it can be classical predicate logic. We construct $\mathsf{T}(\mathsf{L})$ by allowing substitution of formulae of L for the atoms of formulae of T. We are *not* allowing the substitution of formulae of T or even formulae of $\mathsf{T}(\mathsf{L})$ for atoms of L. Thus the temporal connectives of T are never within the scope of connectives of L.

Next we define $\mathsf{T}(\mathsf{L})$ both syntactically and semantically and we prove soundness and completeness for $\mathsf{T}(\mathsf{L})$.

2.2.1 Temporalising a Logic System

Having defined a family of US-temporal logics in Section 1.2, we now externally apply such logic systems to any other logic system L, i.e. we "temporalise" L.

We constrain the logic system L to be an extension of classical logic, i.e. all propositional tautologies must be valid in it. This constraint is due to the fact that all US-temporal logics presented above are extensions of classical logic and any of

them can be taken as the logic T in which we base the temporalisation. We discuss later in this section what should be the case if L is not an extension of classical logic.

Definition 2.1 Boolean combinations and monolithic formulae The set $\mathcal{L}_{\mathsf{L}}$ is partitioned in two sets, BC_{L} and ML_{L}. A formula $A \in \mathcal{L}_{\mathsf{L}}$ belongs to the set of *boolean combinations*, BC_{L}, iff it is built up from other formulae by the use of one of the boolean connectives $\neg$ or $\wedge$ or any other connective defined only in terms of those; it belongs to the set of *monolithic formula* ML_{L} otherwise. □

We can proceed then to the definition of the temporalised language. In the following we will use α, β, γ, ... , to range over formulae of $\mathsf{T}(\mathsf{L})$.

The result of temporalising over $\mathcal{K}$ the logic system L is the logic system $\mathsf{T}(\mathsf{L})$ $= \langle \mathcal{L}_{\mathsf{T}(\mathsf{L})}, \vdash_{\mathsf{T}(\mathsf{L})} \rangle$ and its class of models over $\mathcal{K}$. The alphabet of the temporalised language uses the alphabet of L plus the two-place operators S and U, if they are not part of the alphabet of L; otherwise, we use $\overline{S}$ and $\overline{U}$ or any other proper renaming.

Definition 2.2 Temporalised formulae The set $\mathcal{L}_{\mathsf{T}(\mathsf{L})}$ of formulae of the logic system L is the smallest set such that:

1. If $\alpha \in ML_{\mathsf{L}}$, then $\alpha \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$;
2. If $\alpha, \beta \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$ then $\neg\alpha \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$ and $(\alpha \wedge \beta) \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$;
3. If $\alpha, \beta \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$ then $S(\alpha, \beta) \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$ and $U(\alpha, \beta) \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$.

The set of *maximal monolithic subformulae of* α, $Mon(\alpha)$, is the set of all monolithic subformulae of α that are used to build α up by the rules above. □

It is obvious from the definition above that the set $\mathcal{L}_{\mathsf{T}(\mathsf{L})}$ is denumerably infinite. Note that from item 1 and 2 of the definition above, it follows that $\mathcal{L}_{\mathsf{L}} \subset \mathcal{L}_{\mathsf{T}(\mathsf{L})}$. The reason to define the base case in item 1 in terms of monolithic formulae of L instead of simply defining it in terms of any formula in $\mathcal{L}_{\mathsf{L}}$ is that we would have a double parsing problem. In fact, suppose that instead of item 1 we had a simpler item $1'$ that would state that:

$1'$. If $\alpha \in \mathcal{L}_{\mathsf{L}}$, then $\alpha \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$.

Suppose we want to define a function over the set of formulae, e.g. the depth of the parsing tree of a formula. Consider the formula $(\alpha \wedge \beta) \in \mathcal{L}_{\mathsf{L}}$; it would belong to $\mathcal{L}_{\mathsf{T}(\mathsf{L})}$ both by items $1'$ and 2. If we parse it by $1'$, then its depth will be 0, but if we parse it by 2, its depth will be 1, i.e. depth is not a well defined function. To

avoid such problem we introduce the restriction to monolithic formulae in item 1. We also note that, for instance, if $\Box$ is an operator of the alphabet of L and α and β are two formulae in $\mathcal{L}_{\mathsf{L}}$, the formula $\Box U(\alpha, \beta)$ is *not* in $\mathcal{L}_{\mathsf{T(L)}}$.

There is nothing to prevent us from defining the temporalisation in terms of some F, P-temporal language, but since the language with S and U is more expressive it has received our preference.

If L is an extension of classical logic, we must pay attention to some details before being able to describe the semantics of $\mathsf{T(L)}$. First, if $\mathcal{M}_{\mathsf{L}}$ is a model in the class of models of L, $\mathcal{K}_{\mathsf{L}}$, for every formula $\alpha \in \mathcal{L}_{\mathsf{L}}$ we must have either $\mathcal{M}_{\mathsf{L}} \models \alpha$ or $\mathcal{M}_{\mathsf{L}} \models \neg\alpha$. For example, if L is a modal logic system, e.g. $\mathsf{S4}$, we must consider a "current world" o as part of its model to achieve that condition. Second, we must be careful about the semantics of boolean connectives in the temporalised system. The construction of temporalised formulae based on monolithic formulae of $\mathcal{L}_{\mathsf{L}}$ guarantees that the semantics of the boolean connectives is the same in both the upper-level temporal logic system T and in the temporalised system $\mathsf{T(L)}$.

The language of $\mathsf{T(L)}$ is independent of the underlying flow of time, but not its semantics and inference system, so we must fix a class $\mathcal{K}$ of flows of time over which the temporalisation is defined; this is equivalent to fixing one logic T among the family of temporal logics presented above.

We are then in a position to define the semantics of the temporalised logic system $\mathsf{T(L)}$. Figure 2.1 and Figure 2.2 give us a good idea of the process of generating a temporalised semantics.

Definition 2.3 Semantics of the temporalised logic Consider a flow of time $(T, <) \in \mathcal{K}$ and a function $g : T \to \mathcal{K}_{\mathsf{L}}$, mapping every time point in T to a model in the class of models of L. A model of $\mathsf{T(L)}$ is a triple $\mathcal{M}_{\mathsf{T(L)}} = (T, <, g)$ and the fact that α is true in $\mathcal{M}_{\mathsf{T(L)}}$ at time t is written as $\mathcal{M}_{\mathsf{T(L)}}, t \models \alpha$ and defined as:

$\mathcal{M}_{\mathsf{T(L)}}, t \models \alpha,\ \alpha \in ML_{\mathsf{L}}$ iff $g(t) = \mathcal{M}_{\mathsf{L}}$ and $\mathcal{M}_{\mathsf{L}} \models \alpha$.

$\mathcal{M}_{\mathsf{T(L)}}, t \models \neg\alpha$ iff it is not the case that $\mathcal{M}_{\mathsf{T(L)}}, t \models \alpha$.

$\mathcal{M}_{\mathsf{T(L)}}, t \models (\alpha \wedge \beta)$ iff $\mathcal{M}_{\mathsf{T(L)}}, t \models \alpha$ and $\mathcal{M}_{\mathsf{T(L)}}, t \models \beta$.

$\mathcal{M}_{\mathsf{T(L)}}, t \models S(\alpha, \beta)$ iff there exists $s \in T$ such that $s < t$ and $\mathcal{M}_{\mathsf{T(L)}}, s \models \alpha$ and for every $u \in T$, if $s < u < t$ then $\mathcal{M}_{\mathsf{T(L)}}, u \models \beta$.

$\mathcal{M}_{\mathsf{T(L)}}, t \models U(\alpha, \beta)$ iff there exists $s \in T$ such that $t < s$ and $\mathcal{M}_{\mathsf{T(L)}}, s \models \alpha$ and for every $u \in T$, if $t < u < s$ then $\mathcal{M}_{\mathsf{T(L)}}, u \models \beta$. $\Box$

We write $\mathsf{T}(\mathsf{L}) \models \alpha$ if, for every model $\mathcal{M}_{\mathsf{T}(\mathsf{L})}$ whose underlying flow of time $(T, <) \in \mathcal{K}$ and for every time point $t \in T$, it is the case that $\mathcal{M}_{\mathsf{T}(\mathsf{L})}, t \models \alpha$.

The inference system of $\mathsf{T}(\mathsf{L})/\mathcal{K}$ is given by the following:

Definition 2.4 Axiomatisation for $\mathsf{T}(\mathsf{L})$

- The axioms of $\mathsf{T}/\mathcal{K}$;
- The inference rules of $\mathsf{T}/\mathcal{K}$;
- For every formula α in $\mathcal{L}_{\mathsf{L}}$, if $\vdash_{\mathsf{L}} \alpha$ then $\vdash_{\mathsf{T}(\mathsf{L})} \alpha$. □

The third item above constitutes a new inference rule needed to preserve the theoremhood of formulae of the logic system L. Therefore we call it **Preserve**. The only inference rules we are considering in this presentation are **Subst**, **MP** and **TG**, but other rules such as the irreflexivity rule **IRR**, [Gabbay and Hodkinson 1990], can also be added.

The first concern about the axiomatisation is its soundness, i.e. if whenever $\vdash_{\mathsf{T}(\mathsf{L})} \alpha$ we have $\mathsf{T}(\mathsf{L}) \models \alpha$.

Theorem 2.1 (Soundness of $\mathsf{T}(\mathsf{L})$) *If the logic system L is sound and $\mathsf{US}/\mathcal{K}$ is sound over the class of flows of time $\mathcal{K}$, then so is the logic system $\mathsf{T}(\mathsf{L})/\mathcal{K}$.*

Proof Soundness of $\mathsf{US}/\mathcal{K}$ gives us the validity of the axioms over $\mathcal{K}$. As for the inference rules, soundness of L guarantees that all formulae generated by **Preserve** are valid; soundness of $\mathsf{US}/\mathcal{K}$ guarantees that the other inference rules, when applied to valid formulae, always generate valid formulae. □

Completeness is discussed later in 2.2.3. Let us first present a few examples of the temporalisation of an existing logic system.

Example 2.1 Temporalising modal logic of belief Suppose we have a propositional modal logic of belief $\mathsf{B} = \langle \mathcal{L}_{\mathsf{B}}, \vdash_{\mathsf{B}} \rangle$ with the modal operator B, in which Bp is intended to mean that p is a proposition that is believed by an agent. The axiomatisation, $\vdash_{\mathsf{B}}$, is given by the basic modal logic system K plus the transitivity axiom **4** as one of the introspective properties of belief systems in [Hintikka 1962]:

$$K \begin{cases} \text{All propositional tautologies} \\ B(p \to q) \to (Bp \to Bq) \\ \text{Rules: } \textbf{Subst}, \textbf{MP}, B\textbf{-necessitation} \end{cases} + \; Bp \to BBp$$

The transitivity axiom means that, if some fact is believed, it is believed to be believed, which represents a positive introspection of the believing agent; for a discussion on modal logics of belief, see [Halpern and Moses 1985]. This system is provided with a standard Kripke semantics for modal logics [Hughes and Cresswell 1968], with a set of possible worlds W, an accessibility relation R and a valuation function V, so that $\mathcal{M}_{\mathsf{B}} = (W, R, V)$ is a model structure in which the accessibility relation R is transitive. Actually, we are considering $\mathcal{M}_{\mathsf{B}} = (W, R, V, o)$, where o is a "current world" from which the observations are made, so that we may have both validity and satisfiability in the model theory of B.

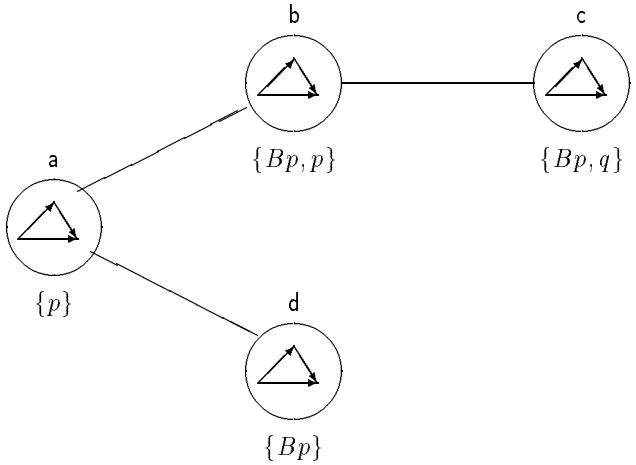
Consider the temporalised logic system $\mathsf{T(B)}$ over the class $\mathcal{K}_0$ of all flows of time. Its inference system $\vdash_{\mathsf{T(B)}}$, for example, gives us as theorems

$$B(p \rightarrow q) \rightarrow (Bp \rightarrow Bq)$$
$$\neg(Bp \wedge \neg Bp)$$
$$GB\neg(Bp \wedge \neg Bp)$$
$$G(Bp \rightarrow q) \rightarrow (U(Bp, Bq) \rightarrow U(q, Bq)).$$

Suppose we have a theory, *i.e.* a set of formulae, $\Gamma = \{GBp, Bp \rightarrow Fp, U(q, Bp)\}$. We construct one possible model $\mathcal{M}_{\mathsf{T(B)}}$ by choosing a flow of time with $T = \{a, b, c, d\}$ and the partial order $< = \{(a, b), (b, c), (a, c), (a, d)\}$. We construct the assignment g such that:

$$g(a) = \mathcal{M}_{\mathsf{B,a}} \models p$$
$$g(b) = \mathcal{M}_{\mathsf{B,b}} \models Bp \wedge p,$$
$$g(c) = \mathcal{M}_{\mathsf{B,c}} \models Bp \wedge q \text{ and}$$
$$g(d) = \mathcal{M}_{\mathsf{B,d}} \models Bp$$

In the resulting model $\mathcal{M}_{\mathsf{T(B)}} = (T, <, g)$, where T, $<$ and g are constructed as above, we have $\mathcal{M}_{\mathsf{T(B)}}, a \models \Gamma$ as illustrated below.



Bp holds in all points to the future of a, $\{b, c, d\}$, so GBp holds at a; Bp is false at a, so $Bp \rightarrow Fp$ holds at a; and q holds at c and Bp holds at b, the only time point between a and c, so $U(q, Bp)$ holds at a. □

Example 2.2 Temporalising propositional logic Consider classical propositional logic $\mathsf{PL} = \langle \mathcal{L}_{\mathsf{PL}}, \vdash_{\mathsf{PL}} \rangle$. Its temporalisation generates the logic system $\mathsf{T(PL)} = \langle \mathcal{L}_{\mathsf{T(PL)}}, \vdash_{\mathsf{T(PL)}} \rangle$.

It is not difficult to see that $\mathcal{L}_{\mathsf{T(PL)}} = \mathcal{L}_{\mathsf{US}}$ and $\vdash_{\mathsf{T(PL)}} = \vdash_{\mathsf{US}}$, i.e. the temporalised version of PL over any $\mathcal{K}$ is actually the temporal logic $\mathsf{T} = \mathsf{US}/\mathcal{K}$. With respect to $\mathcal{M}_{\mathsf{T(L)}}$, the function g actually assigns, for every time point, a PL model. □

Example 2.3 Temporalising US-temporal logic If we temporalise over $\mathcal{K}$ the one-dimensional logic system $\mathsf{US}/\mathcal{K}$ we obtain the two-dimensional logic system $\mathsf{T(US)} = \langle \mathcal{L}_{\mathsf{T(US)}}, \vdash_{\mathsf{T(US)}} \rangle = \mathsf{T}^2\mathsf{(PL)}/\mathcal{K}$. In this case we have to rename the two-place operators S and U of the temporalised alphabet to, say, $\overline{S}$ and $\overline{U}$.

In order to obtain a model for $\mathsf{T(US)}$, we must fix a "current time", o, in $\mathcal{M}_{\mathsf{US}} = (T_1, <_1, g_1)$, so that we can construct the model $\mathcal{M}_{\mathsf{T(US)}} = (T_2, <_2, g_2)$ as previously described. Note that, in this case, the flows of time $(T_1, <_1)$ and $(T_2, <_2)$ need not to be the same. $(T_2, <_2)$ is the flow of time of the upper-level temporal system whereas $(T_1, <_1)$ is the flow of time of the underlying logic which, in this case, happens to be a temporal logic.

The logic system we obtain by temporalising US-temporal logic is the two-dimensional temporal logic described in [Finger 1992]. □

Example 2.4 N-dimensional temporal logic If we repeat the process started in the last two examples, we can construct an n-dimensional temporal logic $\mathsf{T}^n\mathsf{(PL)}/\mathcal{K}$ (its alphabet including S_n and U_n) by temporalising a $(n-1)$-dimensional temporal logic.

Every time we add a temporal dimension, we are able to describe changes in the underlying system. Temporalising the system L once, we are creating a way of describing the history of L; temporalising for the second time, we are describing how the history of L is viewed in different moments of time. We can go on indefinitely, although it is not clear what is the purpose of doing so. □

The assumption that the underlying logic system L is an extension of classical logic allows us to make a clear distinction between boolean and monolithic formulae, avoiding double parsing and reconstructing the boolean formulae and its semantics in the temporalised system $\mathsf{T(L)}$. If we were to temporalise a logic system that is

not an extension of classical logic we could consider all its formulae as being monolithic.The problem would then be the different semantics of the boolean connectives in the underlying system and in the upper-level (classical) temporal system, if those symbols are identical in both systems. The solution would be renaming the boolean connectives, say, in the underlying system. The applications of such a hybrid logic system are not clear so, to avoid extra difficulties in the results we are going to prove, we will stick to the constraint that L is an extension of classical logic.

2.2.2 The Correspondence Mapping

We are now going to relate the temporalised logic system $\mathsf{T(L)}$ to the original US-temporal logic used as a base for the temporalisation process. Consider $\mathcal{P}$, a denumerably infinite set of propositional letters, and let US be the propositional temporal logic system induced by $\mathcal{P}$. The following defines a relationship between a temporalised language $\mathcal{L}_{\mathsf{T(L)}}$ and a propositional temporal language $\mathcal{L}_{\mathsf{US}}$.

Definition 2.5 The correspondence mapping Consider an enumeration p_1, p_2, ..., of elements of $\mathcal{P}$ and consider an enumeration α_1, α_2, ..., of formulae in ML_{L}. The *correspondence mapping* $\sigma : \mathcal{L}_{\mathsf{T(L)}} \rightarrow \mathcal{L}_{\mathsf{US}}$ is given by:

$$\begin{aligned}
\sigma(\alpha_i) &= p_i \text{ for every } \alpha_i \in ML_{\mathsf{L}}, i = 1, 2 \ldots \\
\sigma(\neg\alpha) &= \neg\sigma(\alpha) \\
\sigma(\alpha \wedge \beta) &= \sigma(\alpha) \wedge \sigma(\beta) \\
\sigma(S(\alpha, \beta)) &= S(\sigma(\alpha), \sigma(\beta)) \\
\sigma(U(\alpha, \beta)) &= U(\sigma(\alpha), \sigma(\beta))
\end{aligned}$$

□

The following is the *correspondence lemma*, linking temporalised formulae to temporal logic ones.

Lemma 2.1 *The correspondence mapping is a bijection*

Proof By two straightforward structural inductions we can prove that σ is both injective and surjective. Details are omitted. □

As a consequence, we can always refer to an element Q of $\mathcal{L}_{\mathsf{US}}$ as $\sigma(\alpha)$, because there is guaranteed to be a unique $\alpha \in \mathcal{L}_{\mathsf{T(L)}}$ such that α is mapped into Q by σ. It is clear that the constant $\bot$ defined as $\bot = p \wedge \neg p$ maps into $\sigma(\bot) = \sigma(p) \wedge \neg\sigma(p) = \bot$. We can then establish a connection between consistent formulae in $\mathsf{T(L)}/\mathcal{K}$ and in $\mathsf{US}/\mathcal{K}$.

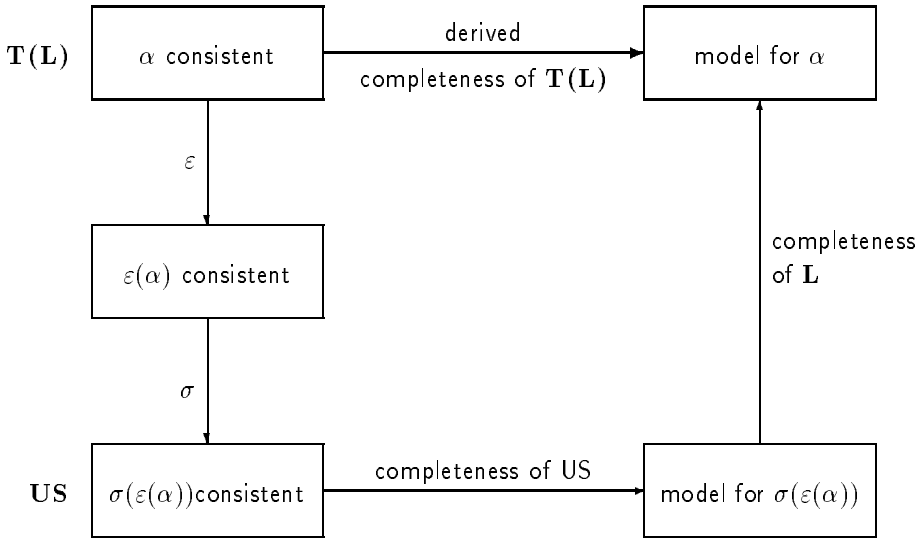


Figure 2.3 Strategy for the proof of completeness

Lemma 2.2 *If* α *is* $\mathsf{T(L)}$*-consistent then* $\sigma(\alpha)$ *is* US*-consistent.*

Proof Suppose $\sigma(\alpha)$ is inconsistent. Since all axioms and inference rules in $\mathsf{US}/\mathcal{K}$ are also in $\mathsf{T(L)}/\mathcal{K}$, the derivation of $\vdash_{\mathsf{US}} \sigma(\alpha) \rightarrow \bot$ can be imitated to derive $\vdash_{\mathsf{T(L)}} \alpha \rightarrow \bot$, which contradicts α being $\mathsf{T(L)}$-consistent. □

The results above are very useful for the proof of completeness and decidability of $\mathsf{T(L)}$.

2.2.3 Completeness of T(L)

We are going to show here that whenever there exists a complete axiomatisation for $\mathsf{US}/\mathcal{K}$ and for L, where $\mathcal{K} \subseteq \mathcal{K}_{lin}$ is any linear class of flows of time, then the temporalised logic system $\mathsf{T(L)}/\mathcal{K}$ is also complete.

The strategy of the completeness proof is illustrated in Figure 2.3. We prove the completeness of $\mathsf{T(L)}/\mathcal{K}$ indirectly by transforming a consistent formula of $\mathsf{T(L)}$ and then mapping it into a consistent formula of US. Completeness of $\mathsf{US}/\mathcal{K}$ is used to find a model for the mapped formula that is used to construct a model for the original $\mathsf{T(L)}$ formula.

The transformation function ε is introduced to deal with the differences between deductions in US and $\mathsf{T(L)}$ due to the presence of the inference rule **Preserve** in $\mathsf{T(L)}$. This inference rule states that theorems in L are also theorems in $\mathsf{T(L)}$. The model theoretic counterpart of this property is that valid formulae in L are also valid in $\mathsf{T(L)}$. The idea behind the transformation ε is to extract the "valid

and contradictory content" that formulae of $\mathsf{T}(\mathsf{L})$ may have due to the validity or unsatisfiability of some set of its subformulae in L.

Definition 2.6 The transformations η and ε Given a formula $\alpha \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$, consider the following sets:

$$\begin{aligned} Lit(\alpha) &= Mon(\alpha) \cup \{\neg\beta \mid \beta \in Mon(\alpha)\} \\ Inc(\alpha) &= \{\textstyle\bigwedge\Gamma \mid \Gamma \subseteq Lit(\alpha) \text{ and } \Gamma \vdash_{\mathsf{L}} \bot\} \end{aligned}$$

where $Mon(\alpha)$ is the set of maximal monolithic subformulae of α. We define then the operator $\Box$ (always) and the formulae $\eta(\alpha)$ and $\varepsilon(\alpha)$:

$$\begin{aligned} \Box\beta &= \beta \wedge G\beta \wedge H\beta \\ \eta(\alpha) &= \bigwedge_{\beta \in Inc(\alpha)} \Box\neg\beta \\ \varepsilon(\alpha) &= \alpha \wedge \eta(\alpha) \end{aligned}$$

□

Since $\eta(\alpha)$ is a theorem of $\mathsf{T}(\mathsf{L})$, we have the following lemma.

Lemma 2.3 $\vdash_{\mathsf{T}(\mathsf{L})} \varepsilon(\alpha) \leftrightarrow \alpha$

If $\mathcal{K}$ is a subclass of linear flows of time, we also have the following property:

Lemma 2.4 *Let $\mathcal{M}_{\mathsf{US}}$ be a temporal model over $\mathcal{K} \subseteq \mathcal{K}_{lin}$ such that for some $o \in T$, $\mathcal{M}_{\mathsf{US}}, o \models \sigma(\Box\alpha)$. Then, for every $t \in T$, $\mathcal{M}_{\mathsf{US}}, t \models \sigma(\Box\alpha)$.*

Therefore, if some subset of $Lit(\alpha)$ is inconsistent, the transformed formula $\varepsilon(\alpha)$ puts that fact in evidence so that, when σ maps it into US, inconsistent subformulae will be mapped into falsity.

To prove the completeness of $\mathsf{T}(\mathsf{L})/\mathcal{K}$ given the completeness of $\mathsf{US}/\mathcal{K}$, we fix an α and assume it is a $\mathsf{T}(\mathsf{L})$-consistent formula. We have then to construct a model for α over $\mathcal{K}$.

By Lemma 2.3, the formula $\varepsilon(\alpha)$ is $\mathsf{T}(\mathsf{L})$-consistent and, by Lemma 2.2, $\sigma(\varepsilon(\alpha))$ is US-consistent. Then, by the completeness of $\mathsf{US}/\mathcal{K}$, there exists a model $\mathcal{M}_{\mathsf{US}} = (T, <, h)$ with $(T, <) \in \mathcal{K}$ such that for some $o \in T$, $\mathcal{M}_{\mathsf{US}}, o \models \sigma(\varepsilon(\alpha))$.

For every $t \in T$, define $G_\alpha(t)$:

$$G_\alpha(t) = \{\beta \in Lit(\alpha) \mid \mathcal{M}_{\mathsf{US}}, t \models \sigma(\beta)\}$$

Lemma 2.5 *If α is $\mathsf{T}(\mathsf{L})$-consistent, then for every $t \in T$, $G_\alpha(t)$ is finite and L-consistent.*

Proof Since $Lit(\alpha)$ is finite, $G_\alpha(t)$ is finite for every t. Suppose $G_\alpha(t)$ is inconsistent for some t, then there exist $\{\beta_1, \ldots, \beta_n\} \subseteq G_\alpha(t)$ such that $\vdash_{\mathsf{L}} \bigwedge \beta_i \rightarrow \bot$. So $\bigwedge \beta_i \in Inc(\alpha)$ and $\Box\neg(\bigwedge \beta_i)$ is one of the conjuncts of $\varepsilon(\alpha)$. Applying Lemma 2.4 to $\mathcal{M}_{\mathsf{US}}, o \models \sigma(\varepsilon(\alpha))$ we get that for every $t \in T$, $\mathcal{M}_{\mathsf{US}}, t \models \neg(\bigwedge \sigma(\beta_i))$ but by, the definition of G_α, $\mathcal{M}_{\mathsf{US}}, t \models \bigwedge \sigma(\beta_i)$, which is a contradiction. □

We are finally ready to prove the completeness of $\mathsf{T}(\mathsf{L})/\mathcal{K}$.

Theorem 2.2 (Completeness for $\mathsf{T}(\mathsf{L})$) *If the logical system L is complete and $\mathsf{US}/\mathcal{K}$ is complete over a subclass of linear flows of time $\mathcal{K} \subseteq \mathcal{K}_{lin}$, then the logical system $\mathsf{T}(\mathsf{L})/\mathcal{K}$ is complete over $\mathcal{K}$.*

Proof Assume that α is $\mathsf{T}(\mathsf{L})$-consistent. By Lemma 2.5, we have $(T, <) \in \mathcal{K}$ and associated to every time point in T we have a finite and L-consistent set $G_\alpha(t)$. By (weak) completeness of L, every $G_\alpha(t)$ has a model, so we define the temporalised valuation function g:

$$g(t) \;=\; \{\mathcal{M}^t_{\mathsf{L}} \mid \mathcal{M}^t_{\mathsf{L}} \text{ is a model of } G_\alpha(t)\}$$

Consider the model $\mathcal{M}_{\mathsf{T}(\mathsf{L})} = (T, <, g)$ over $\mathcal{K}$. By structural induction over β, we show that for every β that is a subformula of α and for every time point t,

$$\mathcal{M}_{\mathsf{US}}, t \models \sigma(\beta) \text{ iff } \mathcal{M}_{\mathsf{T}(\mathsf{L})}, t \models \beta$$

We show only the basic case, $\beta \in Mon(\alpha)$. Suppose $\mathcal{M}_{\mathsf{US}}, t \models \sigma(\beta)$; then $\beta \in G_\alpha(t)$ and $\mathcal{M}^t_{\mathsf{L}} \models \beta$, and hence $\mathcal{M}_{\mathsf{T}(\mathsf{L})}, t \models \beta$. Suppose $\mathcal{M}_{\mathsf{T}(\mathsf{L})}, t \models \beta$ and assume $\mathcal{M}_{\mathsf{US}}, t \models \neg\sigma(\beta)$; then $\neg\beta \in G_\alpha(t)$ and $\mathcal{M}^t_{\mathsf{L}} \models \neg\beta$, which contradicts $\mathcal{M}_{\mathsf{T}(\mathsf{L})}, t \models \beta$; hence $\mathcal{M}_{\mathsf{US}}, t \models \sigma(\beta)$. The inductive cases are straightforward and details are omitted.

So, $\mathcal{M}_{\mathsf{T}(\mathsf{L})}$ is a model for α over $\mathcal{K}$ and the proof is finished. □

Theorem 2.2 gives us sound and complete axiomatisations for $\mathsf{T}(\mathsf{L})$ over many interesting classes of flows of time, such as the class of all linear flows of time, $\mathcal{K}_{lin}$, the integers, $\mathbb{Z}$, and the reals, $\mathbb{R}$. These classes are, in their US versions, decidable and the corresponding decidability of $\mathsf{T}(\mathsf{L})$ is dealt in Section 2.3. Integer and real flows of time also have the separation property, which is discussed in Section 2.5.

2.3 The Decidability of T(L) and its Complexity

The main goal of this section is to show that, if the logic system L is decidable and the logic system US is decidable over $\mathcal{K} \subseteq \mathcal{K}_{lin}$, then the logic system T(L) is also decidable over $\mathcal{K}$. We assume throughout this section that US/$\mathcal{K}$ is complete.

Definition 2.7 Decidability of a Logic System A logic system L is said to be *decidable* if there exists an algorithm (a decision procedure) that, for every formula $\alpha \in \mathcal{L}_{\mathsf{L}}$, outputs "yes" if α is a theorem in the logic system L and "no" otherwise. □

There are results for decidability of US over several linear classes of flows of time, among which are the class $\mathcal{K}_{lin}$ of all linear flows of time [Burgess 1984], the integer and the real flows of time, [Burgess and Gurevich 1985].

As in the proof of completeness, we are going to prove the decidability result using the correspondence mapping σ and the transformation η. Recall Definition 2.6, in which the sets $Mon(\alpha)$, $Lit(\alpha)$ and $Inc(\alpha)$ were all finite, so that we have the following result about $\eta(\alpha)$.

Lemma 2.6 *For any $\alpha \in \mathcal{L}_{\mathsf{T(L)}}$, if the logic system L is decidable then there exists an algorithm for constructing $\eta(\alpha)$.*

The relationship between T(L) and US that we need to prove the decidability of T(L) is the following:

Lemma 2.7 *Over a linear flow of time, for every $\alpha \in \mathcal{L}_{\mathsf{T(L)}}$,*

$$\vdash_{\mathsf{T(L)}} \alpha \text{ iff } \vdash_{\mathsf{US}} \sigma(\eta(\alpha) \rightarrow \alpha).$$

Proof The *if* case comes trivially from the definition of $\vdash_{\mathsf{T(L)}}$. For the *only if* part, suppose $\vdash_{\mathsf{T(L)}} \alpha$. We prove by induction on the deduction of α that $\vdash_{\mathsf{US}} \sigma(\eta(\alpha) \rightarrow \alpha)$.

Basic cases:

1. α is obtained using the inference rule **Preserve**. Then $\eta(\alpha) = \neg\neg\alpha$ and $\vdash_{\mathsf{US}} \sigma(\neg\neg\alpha \rightarrow \alpha)$.

2. α is obtained using the inference rule **Subst**. Suppose α was obtained by substituting p_i by β_i in some axiom **A**. Then $\vdash_{\mathsf{US}} \alpha$ can be obtained by substituting $\sigma(p_i)$ by $\sigma(\beta_i)$ in axiom **A**.

Inductive cases:

1. $\alpha = G\beta$ is obtained using the inference rule **TG**. Note that $\eta(\alpha) = \eta(\beta)$. Then
   $\vdash_{\mathsf{US}} \sigma(\eta(\alpha)) \rightarrow \sigma(\beta)$ by induction hypothesis
   $\vdash_{\mathsf{US}} G(\sigma(\eta(\alpha)) \rightarrow \sigma(\beta))$ by **TG**
   $\vdash_{\mathsf{US}} G(\sigma(\eta(\alpha))) \rightarrow \sigma(\alpha)$ by temporal logic and $\alpha = G\beta$
   $\vdash_{\mathsf{US}} \sigma(\eta(\alpha)) \rightarrow G(\sigma(\eta(\alpha)))$ by the definition of η and $\mathcal{K}$ linear
   $\vdash_{\mathsf{US}} \sigma(\eta(\alpha) \rightarrow \alpha)$ from the two previous lines
   Similarly for $\alpha = H\beta$.

2. α is obtained from β and $\beta \rightarrow \alpha$ by **MP**. Then
   $\vdash_{\mathsf{US}} \sigma(\eta(\beta)) \rightarrow \sigma(\beta)$ by induction hypothesis
   $\vdash_{\mathsf{US}} \sigma(\eta(\beta \rightarrow \alpha)) \rightarrow \sigma(\beta \rightarrow \alpha)$ by induction hypothesis
   $\vdash_{\mathsf{US}} \sigma(\eta(\beta \rightarrow \alpha)) \rightarrow \sigma(\eta(\beta))$ by the definition of η
   $\vdash_{\mathsf{US}} \sigma(\eta(\beta \rightarrow \alpha)) \rightarrow \sigma(\beta)$ from the $3rd$ and $1st$ lines
   $\vdash_{\mathsf{US}} \sigma(\eta(\beta \rightarrow \alpha)) \rightarrow \sigma(\alpha)$ from the $4th$ and $2nd$ lines
   Let p be a proposition that occurs in $\sigma(\beta)$ but not in $\sigma(\alpha)$. If we eliminate from $\sigma(\eta(\alpha \rightarrow \beta))$ all the conjuncts in which p occurs, obtaining $\sigma(\gamma)$, using the completeness of $\mathsf{US}/\mathcal{K}$ we can get $\vdash_{\mathsf{US}} \sigma(\gamma) \rightarrow \sigma(\alpha)$. If we do that for all such propositions, we end up with $\vdash_{\mathsf{US}} \sigma(\eta(\alpha) \rightarrow \alpha)$. □

Theorem 2.3 (Decidability of $\mathsf{T}(\mathsf{L})$) *If* L *is a decidable logic system, and* US *is decidable over* $\mathcal{K} \subseteq \mathcal{K}_{lin}$*, then the logic system* $\mathsf{T}(\mathsf{L})$ *is also decidable over* $\mathcal{K}$*.*

Proof Consider $\alpha \in \mathcal{L}_{\mathsf{T}(\mathsf{L})}$. Since L is decidable, by Lemma 2.6 there is an algorithmic procedure to build $\eta(\alpha)$. Since σ is a recursive function, we have an algorithm to construct $\sigma(\eta(\alpha) \rightarrow \alpha)$, and due to the decidability of US over $\mathcal{K}$, we have an effective procedure to decide if it is a theorem or not. Since $\mathcal{K}$ is linear, by Lemma 2.7 this is also a procedure for deciding whether α is a theorem or not. □

Once we have a decidability result, the next natural question is about the complexity of the decision procedure. We briefly discuss here an upper bound for the complexity analysis. Let N be the number of (boolean and modal) connectives in a formula, let the complexity of the decision procedure in L be $\mathcal{O}(f_{\mathsf{L}}(N))$ and in US be $\mathcal{O}(f_{\mathsf{US}}(N))$. The decision procedure for $\mathsf{T}(\mathsf{L})$ as given by the proof above consists of basically two steps:

1. constructing $\eta(\alpha)$;

2. deciding whether $\sigma(\eta(\alpha) \rightarrow \alpha)$ is a theorem or not;

The construction of $\eta(\alpha)$ involves generating all subsets of $Lit(\alpha)$ and applying the decision procedure for each subset, therefore its complexity is $\mathcal{O}(2^N \times f_{\mathsf{L}}(N))$. The second step is dominated by the decision procedure of US since the application of σ can be done in polynomial time; in the worst case, when all tests in L succeed, the size of $\eta(\alpha)$ is $\mathcal{O}(2^N)$ and therefore the decision is $\mathcal{O}(f_{\mathsf{US}}(2^N))$. So an upper bound for the decision procedure for T(L) is given by the dominating term of $\mathcal{O}(2^N \times f_{\mathsf{L}}(N))$ and $\mathcal{O}(f_{\mathsf{US}}(2^N))$. As for a lower bound for the decision procedure of T(L), it cannot be any lower than the highest of the lower bounds for US and L.

2.4 Conservativeness of T(L)

Conservativeness can be easily derived from the soundness of US and the completeness of L, without any assumptions on the flow of time.

Definition 2.8 Conservative extension A logic system L_1 is an *extension* of a logic system L_2 if $\mathcal{L}_{\mathsf{L}_2} \subseteq \mathcal{L}_{\mathsf{L}_1}$ and if $\vdash_{\mathsf{L}_2} \alpha$ then $\vdash_{\mathsf{L}_1} \alpha$. A logic L_1 is a *conservative extension* of L_2 if it is an extension of L_2 such that if $\alpha \in \mathcal{L}_{\mathsf{L}_2}$, then $\vdash_{\mathsf{L}_1} \alpha$ only if $\vdash_{\mathsf{L}_2} \alpha$. □

We know that all complete US are conservative extensions of predicate logic PL. Clearly, T(L) is an extension of L. We prove that it is also conservative.

Theorem 2.4 (Conservativeness of T(L)) *Let* L *be a complete logic system and* US *be sound over* $\mathcal{K}$. *The logic system* T(L) *is a conservative extension of* L.

Proof Let $\alpha \in \mathcal{L}_{\mathsf{L}}$ such that $\vdash_{\mathsf{T(L)}} \alpha$. Suppose for contradiction that $\nvdash_{\mathsf{L}} \alpha$, so by completeness of L, there exists a model $\mathcal{M}_{\mathsf{L}}$ such that $\mathcal{M}_{\mathsf{L}} \models \neg\alpha$. We construct a temporalised model $\mathcal{M}_{\mathsf{T(L)}} = (T, <, g)$ by making $g(t) = \mathcal{M}_{\mathsf{L}}$ for all $t \in T$. $\mathcal{M}_{\mathsf{T(L)}}$ clearly contradicts the soundness of T(L) and therefore that of US, so $\vdash_{\mathsf{L}} \alpha$. □

2.5 Separation over the Added Dimension

The separation property of the US-temporal logic allows us to rewrite any temporal formula into a conjunction of formulae of the form

$$\text{past formula and present formula} \rightarrow \text{future formula.}$$

Once a formula is in the format above, it can be imperatively interpreted against a partial temporal model according to [Gabbay 1987], so that if the antecedent holds

in the past and present in the model, then we must execute the consequent in the future so as to make the formula true in the model. The imperative interpretation of a formula (also called the execution of a temporal specification) is based on an asymmetric view of the flow of time; in a symmetric view of time, whenever the antecedent is true in the past and present, we could either make the consequent true in the future or we could try to falsify the antecedent itself, in both cases maintaining the validity of the temporal specification. In this asymmetric view of time, we discard the latter possibility and remain with the former as the only possibility for the execution of a temporal specification.

In this section we want to extend this imperative interpretation of a temporal formula over a logic system L so that, after temporalising L over a flow of time that is like the integers or reals, we can execute temporal specifications in T(L). The concept of a separated formula is based on the notion of a pure formula, so we present the definitions of pure formula and separated formula for the US logic.

Definition 2.9 Pure formulae in US

1. A *pure present formula* is a boolean combination of propositional letters.

2. A *pure past formula* is a boolean combination of formulae of the form $S(\alpha, \beta)$ where α and β are either pure present or pure past formulae.

3. A *pure future formula* is a boolean combination of formulae of the form $U(\alpha, \beta)$ where α and β are either pure present or pure future formulae.

A *separated formula* is a formula that is a boolean combination of pure formulae only. □

Once we have a separated formula, it can be brought to a conjunctive normal form, i.e. a conjunction of disjuncts, so that each conjunct can be finally brought to the form:

$$\text{pure-present and pure-past} \rightarrow \text{pure-future.}$$

The following is the basic result about separation over the integers.

Theorem 2.5 (Separation Theorem) *For any formula $A \in \mathcal{L}_{\mathsf{US}}$ there exists a separated formula $B \in \mathcal{L}_{\mathsf{US}}$ such that A is equivalent to B over an integer-like flow of time.*

A proof of the separation theorem can be found in [Gabbay 1987; Gabbay, Hodkinson and Reynolds 1994]. It also holds for the reals. Unfortunately, given a formula A, the automatic generation of an equivalent separated formula involves exponential computations.

The generalisation of pure formula for a temporalised logic system $\mathsf{T}(\mathsf{L})$ is given below.

Definition 2.10 Pure temporalised formulae

1. every formula $\alpha \in \mathcal{L}_{\mathsf{L}}$ is a *pure present temporalised formula*.

2. A *pure past temporalised formula* is a boolean combination of formulae of the form $S(\alpha, \beta)$ where α and β are either pure present or pure past temporalised formulae.

3. A *pure future temporalised formula* is a boolean combination of formulae of the form $U(\alpha, \beta)$ where α and β are either pure present or pure future temporalised formulae.

A *separated temporalised formula* is a boolean combination of pure formulae of $\mathsf{T}(\mathsf{L})$. □

Example 2.5 Temporalising a modal logic of belief Suppose L is the modal logic system of belief, with the modal operator B. Here are some examples of pure temporalised formulae in $\mathsf{T}(\mathsf{L})$:

1. Pure present: $Bp \rightarrow p$, $\neg(p \wedge \neg p)$, and any other formula of the logic L.

2. Pure past: $P(Bp) \rightarrow S(Bp, \neg p)$.

3. Pure future: $F(Bp) \rightarrow \neg Fp \vee G(Bp \rightarrow \neg p)$. □

In order to prove the separation theorem for the temporalised logic $\mathsf{T}(\mathsf{L})$ we will use the correspondence mapping. The basic strategy of the proof is illustrated in figure 2.4.

The following is a helpful result that will lead us to the proof of separation for the temporalised logic $\mathsf{T}(\mathsf{L})$.

Lemma 2.8 *Let σ be a correspondence mapping between $\mathcal{L}_{\mathsf{T}(\mathsf{L})}$ and $\mathcal{L}_{\mathsf{US}}$. $\sigma(\alpha)$ is a separated formula in the logic US iff α is a separated formula in $\mathsf{T}(\mathsf{L})$.*

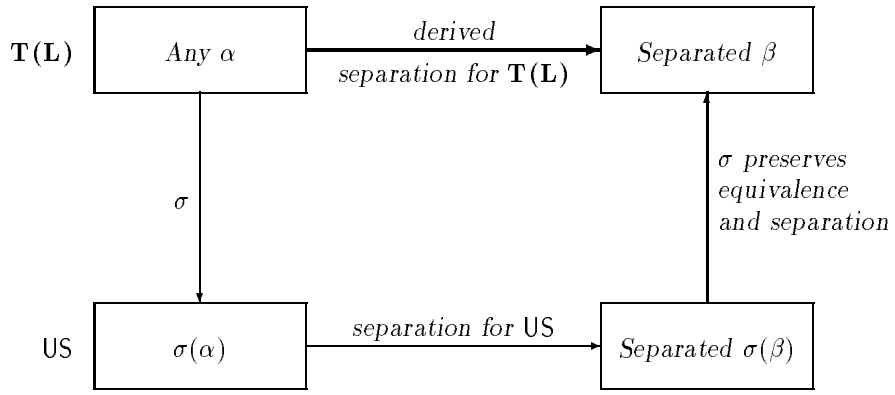


Figure 2.4 Separation of T(L)-formulae via separation of US-formulae

Proof From the definition of the correspondence mapping it follows that if α is a boolean combination of $\alpha_1, \ldots \alpha_n \in \mathcal{L}_{\mathsf{T(L)}}$ then $\sigma(\alpha)$ is a boolean combination of $\sigma(\alpha_1), \ldots, \sigma(\alpha) \in \mathcal{L}_{\mathsf{US}}$. The converse is also true since σ is a bijection.

Therefore, to show that α is separated in T(L) iff $\sigma(\alpha)$ is separated in US, all we have to do is to prove that $\sigma(\alpha)$ is a pure formula iff α is a pure formula. We show the proof for the *only if* case; the *if* part is completely analogous.

Suppose $\sigma(\alpha)$ is a pure present, then it is a boolean combination of propositional letters. Therefore α is a boolean combination of monolithic formulae of L, so α is a formula of L and pure present in T(L).

Suppose $\sigma(\alpha)$ is pure past, then it is a boolean combination of formulae in $\mathcal{L}_{\mathsf{US}}$ of the form $S(\sigma(\beta), \sigma(\gamma))$ where $\sigma(\beta)$ and $\sigma(\gamma)$ are pure present or pure past. Therefore α must be a boolean combination of formulae in $\mathcal{L}_{\mathsf{T(L)}}$ of the form $S(\gamma, \delta)$, where γ and δ are, by induction hypothesis, either pure present or pure past. Therefore α is a pure past formula in $\mathcal{L}_{\mathsf{T(L)}}$.

Suppose $\sigma(\alpha)$ is pure future, then by an argument analogous to the previous case, α is a pure future formula. Therefore we have proved that if $\sigma(\alpha)$ is a pure formula in $\mathcal{L}_{\mathsf{US}}$, α is a pure formula in $\mathcal{L}_{\mathsf{T(L)}}$. □

Theorem 2.6 (Separation Theorem for T(L)) *If α is any formula in $\mathcal{L}_{\mathsf{T(L)}}$, then there exists a separated formula $\beta \in \mathcal{L}_{\mathsf{T(L)}}$ such that β is equivalent to α over an integer-like flow of time.*

Proof All we have to do is to prove that if α and β are formulae of T(L) and $\vdash_{\mathsf{US}} \sigma(\alpha) \leftrightarrow \sigma(\beta)$ then $\vdash_{\mathsf{T(L)}} \alpha \leftrightarrow \beta$. In fact, since all axioms and inference rules of US also belong to T(L), the deduction of $\vdash_{\mathsf{US}} \sigma(\alpha) \leftrightarrow \sigma(\beta)$ also leads to $\vdash_{\mathsf{T(L)}} \alpha \leftrightarrow \beta$.

Let then α be any formula of $\mathsf{T}(\mathsf{L})$. From the separation theorem of US, we know that there exist a separated β, such that $\vdash_{\mathsf{US}} \sigma(\alpha) \leftrightarrow \sigma(\beta)$ and $\sigma(\beta)$ is separated. So by Lemma 2.8, β is also a separated formula equivalent to α. □

Once we have the separation property for the temporalised system $\mathsf{T}(\mathsf{L})$, we can rewrite any temporalised formula into a separated equivalent one of the form

$$\text{pure temporalised past and present} \rightarrow \text{pure temporalised future.}$$

The imperative interpretation of such a formula is the following. If the antecedent holds in past and present models of the logic system L, then we execute the temporalised formula by constructing a future model (or a series or future models) of L so as to make the consequent true.

Since the separation property also holds for a real flow of time, the proof above can be trivially adapted to a real flow of time. Note that the separation property for the temporalised system was obtained without any assumptions on the underlying logic system L, as opposed to the results of soundness, completeness and decidability, all of which depend on whether the property holds for the underlying logic system L.

2.6 Temporalising First-Order Logic

In this section we examine in more detail the addition of a temporal dimension to a first-order language as a particularly interesting application of the temporalisation process. For fully quantified temporal predicate logic over $\mathbb{Z}$,the set of valid formulae is not recursively enumerable [Garson 1984], so it is neither complete nor decidable, despite the fact that $\mathsf{US}/\mathbb{Z}$ is both complete and decidable [Reynolds 1992; Burgess and Gurevich 1985]. It is clear that the discussion of temporalising first-order logic will involve only sublanguages of fully quantified temporal predicate logic.

We will consider a first-order language with the quantifier $\forall$, an equality symbol $=$, a countable set of variables $X = \{x_1, x_2, \ldots\}$, a countable set of predicate symbols $P = \{p_1, p_2, \ldots\}$ such that every predicate symbol has an associated natural number $n > 0$, called its arity, a set C of constant symbols and a set F of functional symbols; C and F are possibly empty. The quantifier $\exists$ can be defined in the normal way as $\exists = \neg\forall\neg$. A term is either a variable, a constant symbol or an n-ary function symbol applied to n terms. The notion of the set of free variables of a formula is the usual one. A sentence is a formula with no free variables.

A first-order domain $\mathcal{D}$ is a non-empty set. An interpretation $\mathcal{I}$ is a mapping that associates, for every constant in the language an element in the domain, and for every n-place predicate symbol an n-ary relation over $\mathcal{D}^n$. An assignment function $\mathcal{A}$ is a mapping that associates every variable with an element of the domain. A first-order model is a pair $\mathcal{M} = (\mathcal{D}, \mathcal{I})$. If t is a term, $[\![t]\!]^{\mathcal{I},\mathcal{A}} \in \mathcal{D}$ represents its extension over the domain $\mathcal{D}$ under interpretation $\mathcal{I}$ and assignment $\mathcal{A}$. The semantics of a first-order language is then defined in the usual way, where $\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \alpha$ reads "$\mathcal{M}$ is a model of the formula α under assignment $\mathcal{A}$":

$$
\begin{array}{ll}
\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} p_i(t_1, \ldots, t_n) & \text{iff } \langle [\![t_1]\!]^{\mathcal{I},\mathcal{A}}, \ldots, [\![t_n]\!]^{\mathcal{I},\mathcal{A}} \rangle \in \mathcal{I}(p_i), \text{ for all} \\
 & \quad n\text{-ary predicate symbols } p_i \in P. \\
\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \neg\alpha & \text{iff } \mathcal{M}, \mathcal{A} \not\models_{\mathbf{FOL}} \alpha. \\
\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \alpha \wedge \beta & \text{iff } \mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \alpha \text{ and } \mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \beta. \\
\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} t_1 = t_2 & \text{iff } [\![t_1]\!]^{\mathcal{I},\mathcal{A}} = [\![t_2]\!]^{\mathcal{I},\mathcal{A}}. \\
\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \forall x \alpha & \text{iff for any assignment } \mathcal{A}' \text{ which agrees} \\
 & \quad \text{with } \mathcal{A}, \text{ except possibly on variable} \\
 & \quad x, \mathcal{M}, \mathcal{A}' \models_{\mathbf{FOL}} \alpha.
\end{array}
$$

We say that α has a model $\mathcal{M}$, and write $\mathcal{M} \models_{\mathbf{FOL}} \alpha$, if $\mathcal{M}, \mathcal{A} \models_{\mathbf{FOL}} \alpha$ for all assignments $\mathcal{A}$ (when α is a sentence, either all assignments or none will satisfy it).

The derivability relation, $\vdash_{\mathbf{FOL}}$, can be any of the existent ones for first-order logic. It can be an axiomatic system, but it needs not.

Since in first-order logic we have a basic distinction between sentences and ordinary formulae, we have to consider both cases of adding a temporal dimension to monolithic sentences and to monolithic formulae in general.

2.6.1 Temporalising First-Order Sentences

If we temporalise first-order sentences, we have no problems in following the methodology we have developed so far. We first identify the monolithic sentences as those that are not in the format $\alpha \wedge \beta$ or $\neg\alpha$. For instance, $\forall x p(x)$ and $\forall x \neg(q(x) \wedge \neg q(x))$ are monolithic sentences, whereas $\exists x p(x)$ (implicit negation) and $\forall x p(x) \wedge \forall y \neg q(y)$ are boolean combinations. We then follow the procedure described in Section 2.2, obtaining the logic system T(FOs). Note that in T(FOs) a temporal operator never occurs inside the scope of a quantifier.

The structure of the first-order models that compose the temporalised model deserves some special attention, since one model may differ form another in several

Element	*Fixed*	*Variable*
Domain	constant domains	variable domains
Constant and Functional Symbols	rigid	non-rigid or flexible
Predicate Symbols	rigid	non-rigid or flexible
Assignment	global	local

Table 2.1 Degrees of freedom in temporalising first-order models.

different ways, as if we had various "degrees of freedom" in generating a temporalised version of first-order models. Those degrees of freedom are illustrated in Table 2.1.

If all first-order models that compose a temporalised model $\mathcal{M}_{\mathsf{T(FOs)}}$ refer to the same domain, a constant domain assumption is satisfied; otherwise, we have varying domains. We may have rigid constant and rigid functional symbols, i.e. they have the same interpretation in every model of the temporalised structure; they are called non-rigid or flexible otherwise. A rigid predicate symbol has the same interpretation at all times; otherwise it is a flexible predicate symbol. And finally, the assignment function may be global, i.e. all variables are assigned the same domain element in all models of the temporalised structure (global assignments make sense only under a constant domain assumption); otherwise, it is a local assignment.

In fact, constant domains or rigid terms or predicates are not a consequence of the temporalisation; they are, actually, further assumptions on the temporalised first-order model made so as to impose some external intended meaning of adding a temporal dimension to a logic system. All the previously established results of soundness, completeness and separation are valid for unconstrained $\mathsf{T(FOs)}$; decidability is obviously not applicable.

Nevertheless, there is no quantification over the temporal operators in $\mathsf{T(FOs)}$, which means that the expressivity of this logic is clearly limited. In the following, we examine one step further in increasing this expressivity, while still keeping the original idea of adding a temporal dimension to a logic system.

2.6.2 Temporalising First-Order Formulae

We take now general monolithic first-order formulae as a basis for the addition of a temporal dimension, i.e. all first-order formulae that are not of the form $\neg\alpha$ or $\alpha \wedge \beta$. We generate thus the logic system $\mathsf{T(FOf)}$. Note that the language of $\mathsf{T(FOs)}$ is contained in the language of $\mathsf{T(FOf)}$.

The particular feature that distinguishes this system from all the previously considered systems is that, since we are considering first-order formulae that may contain free variables, monolithic formulae with free variables only have a defined semantics over a first-order model $\mathcal{M}_{\mathsf{FOL}}$ if a variable assignment function is provided, and the free variables of a first-order formula used to build a temporalised formula α remain free in α.

Therefore, while constructing a model for the system $\mathsf{T(FOf)}$, we must consider the existence of a *global assignment function*, $\mathcal{A}_g$, to cope with the free variables. A global assignment function makes sense only in a constant domain context, so we must have this assumption as well; we further assume that all terms are rigid. The effect of the global assignment $\mathcal{A}_g$ is to ground all the free variables of a temporalised formula α. Only the interpretation of predicate symbols changes among the models of L in the temporalised model structure. We write

$$\mathcal{M}_{\mathsf{T(FOf)}} \models \alpha \quad \text{iff} \quad \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}_g \models \alpha \text{ for every } \mathcal{A}_g.$$

Since the construction of its temporalised model and inference system does not follow exactly the way other temporal systems were constructed, the results previously established of soundness, completeness and separation cannot be applied directly.

We know that the more expressive full first-order temporal logic has no possible finite axiomatisation over several useful classes of linear flows of time like $\{\mathbb{R}\}$, $\{\mathbb{Z}\}$ and $\{\mathbb{N}\}$, e.g. see [Garson 1984], but we do have a finite axiomatisation for $\mathsf{T(FOs)}$; the full first-order temporal logic, or another restriction of it, will be revisited in Chapter 4 in the definition of a query language for temporal databases. The logic system $\mathsf{T(FOf)}$ has an intermediary expressive power and it can be shown that $\mathsf{T(FOf)}$ cannot be finitely axiomatised over the natural numbers, although we will not do it here; it follows that completeness results are not transferred to $\mathsf{T(FOf)}$. Perhaps more interesting is that separation can be achieved for this logic through model theory.

Since the concept of separated formula is purely syntactic and does not depend on the model or the inference system, the definition of a separated temporalised

formulae given by Definition 2.10 is also valid for $\mathsf{T(FOf)}$. For the same reasons, the definition of a correspondence mapping σ and the correspondence lemma 2.1 stating that σ is a bijection are also valid in $\mathsf{T(FOf)}$.

Definition 2.11 Corresponding Model Let $\mathcal{M}_{\mathsf{T(FOf)}} = (T, <, g)$ be a model of $\mathsf{T(FOf)}$, and let $\mathcal{A}$ be a global assignment. We construct the valuation function g_σ such that, for every time point $t \in T$ and for every propositional letter $p = \sigma(\alpha) \in \mathcal{P}$ we have

$$\sigma(\alpha) \in g_\sigma(t) \text{ iff } \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \alpha.$$

A model of the temporal logic system US, $\mathcal{M}^\sigma_{\mathsf{US}} = (T, <, g_\sigma)$, is then called the *corresponding model* of $\mathcal{M}_{\mathsf{T(L)}}$ under the corresponding mapping σ and assignment $\mathcal{A}$. □

Lemma 2.9 *If $\mathcal{M}^\sigma_{\mathsf{US}}$ is the corresponding model of $\mathcal{M}_{\mathsf{T(FOf)}}$ under σ and $\mathcal{A}$ then*

$$\mathcal{M}^\sigma_{\mathsf{US}}, t \models \sigma(\alpha) \textit{ iff } \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \alpha$$

for every $\alpha \in \mathcal{L}_{\mathsf{T(L)}}$ and for every $t \in T$.

Proof Straightforward by structural induction on α. □

Theorem 2.7 (Separation for $\mathsf{T(FOf)}$) *For every $\alpha \in \mathcal{L}_{\mathsf{T(FOf)}}$ there exists a separated formula $\beta \in \mathcal{L}_{\mathsf{T(FOf)}}$ such that β is equivalent to α over an integer-like flow of time.*

Proof Let σ be a correspondence mapping and $\mathcal{A}$ an arbitrary global assignment . Consider a temporalised model $\mathcal{M}_{\mathsf{T(FOf)}} = (T, <, h)$, $(T, <) \in \mathbb{Z}$, and let $\mathcal{M}^\sigma_{\mathsf{US}} = (T, <, g_\sigma)$ be its correspondent model under σ and $\mathcal{A}$. By Lemma 2.9, we have

$$\mathcal{M}^\sigma_{\mathsf{US}}, t \models \sigma(\alpha) \text{ iff } \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \alpha \tag{2.1}$$

for every $\alpha \in \mathcal{L}_{\mathsf{T(FOf)}}$ and for every $t \in T$.

By the separation theorem for US we get that, for every formula $\sigma(\alpha) \in \mathcal{L}_{\mathsf{US}}$ there exists a separated formula $\sigma(\beta) \in \mathcal{L}_{\mathsf{US}}$ such that

$$\mathcal{M}^\sigma_{\mathsf{US}}, t \models \sigma(\alpha) \text{ iff } \mathcal{M}^\sigma_{\mathsf{US}}, t \models \sigma(\beta) \tag{2.2}$$

for all time points $t \in T$.

By Lemma 2.8, we have that the corresponding mapping preserves separation, i.e. β is a separated formula iff $\sigma(\beta)$ is a separated formula and, by application of (2.1)

$$\mathcal{M}^\sigma_{\mathsf{US}}, t \models \sigma(\beta) \text{ iff } \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \beta \tag{2.3}$$

for all time points $t \in T$.

Combining (2.1), (2.2) and (2.3) we get that, for every $\alpha \in \mathcal{L}_{\mathsf{T(FOf)}}$ there exists a separated $\beta \in \mathcal{L}_{\mathsf{T(FOf)}}$ such that, for all $t \in T$

$$\mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \alpha \text{ iff } \mathcal{M}_{\mathsf{T(FOf)}}, \mathcal{A}, t \models \beta \tag{2.4}$$

Since the assignment $\mathcal{A}$ was arbitrarily chosen and the separated β does not depend on the particular choice of $\mathcal{A}$, expression (2.4) holds for any global assignment $\mathcal{A}$, and separation for $\mathsf{T(FOf)}$ remains proved. □

We note that if we fix a current time, o, and a global assignment $\mathcal{A}_g$, we can apply the temporalisation process to the logic system $\mathsf{T(FOf)}$, obtaining a two-dimensional temporal predicate system, $\mathsf{T}^2(\mathsf{FOf})$, as a predicate version of the two-dimensional propositional system described in example 2.3.

2.7 Internalising the Temporal Dimension

There are three basic approaches to adding a temporal dimension to a logic system, namely:

1. The temporal operators approach, *i.e.* the external approach.
2. The first-order internalisation of the temporal dimension.
3. A mixed approach combining the two approaches above.

Those three different approaches are discussed in detail in [Gabbay 1990] in the context of propositional temporal logic. The first approach is the one we have been following so far. Here we briefly present the other ones in the context of temporalised formulae.

Consider the propositional temporal formula in US

$$\neg raining \rightarrow F\ raining$$

expressing that if it is not raining now, it will rain in the future. This statement could actually be completely coded in a monadic first-order language with a single temporal argument for the predicate *raining*. The resulting formulation would be

$$\neg raining^*(t) \rightarrow \exists s(t < s \wedge raining^*(s)).$$

This process of getting rid of the temporal operators by adding a new temporal argument to the predicates plus some extra conditions on those arguments can be

done systematically by an *internalisation function* $*$ defined inductively over the structure of a formula of T and also taking as argument a reference time point, generating a monadic predicate formula, one sort over time and the other sort over domain elements. We call this process the *internalisation* of the temporal dimension. The internalisation of the temporal dimension can be generalised to temporalised first-order sentences and is basically obtained by the standard translation of temporal logic into predicate logic, e.g. [van Benthem 1983], with an extra argument to incorporate the temporal reference; this extra argument can be interpreted as the result of Quine's "eternalisation" of first-order sentences [Quine 1960].

In the internalised version it is necessary to incorporate a theory expressing the properties of the flow of time $\mathcal{K} = (T, <)$ to restore the deductive capability of temporal formulae. However, there are several flows of time over which there are complete temporal axiomatisations that are not definable in first-order logic, e.g. the integers and the reals.

Another way of getting to a first-order predicate logic approach to temporal logic, as proposed by Gabbay [1991b], is by mixing two predicate logic languages in the following way. Let G (for global) and L (for local) be two first-order languages. The two-sorted predicate language $\mathsf{L}_k^*(\mathsf{G})$ is the result of mixing the G and L (in our present notation it would be $\mathsf{G}(\mathsf{L}_k^*)$). If we consider the language $\mathsf{L}_1^*(\mathsf{G})$, then a formula of the form $A^*(t, x_1, \ldots, x_n)$ means that $A(x_1, \ldots, x_n)$ holds at time t (here the formula $A(x_1, \ldots, x_n)$indicates that A is a first-order formula with free variables $x_1, \ldots, x_n$). This language is the same language of the internalised temporal dimension system. But this approach gives us a way of creating an internalised logic system in a very similar way to that in which a temporalised system was created, i.e. as a result of putting two languages together. In fact, the original languages G and L can be seen as two linked languages "sharing variables' in the language $\mathsf{L}_1^*(\mathsf{G})$ [Gabbay 1991b]. One of the original languages, G, has the exclusivity of dealing with temporal facts, as the upper-level US-temporal logic system, whereas the language L is responsible only for the local behaviour at each point in the flow of time.

The temporal operators approach to a temporalised formula can be seen as treating time points implicitly, always referring to a current time. The first-order internalisation refers explicitly to the points in the flow of time. A hybrid form of internalisation of the temporal dimension can be obtained by combining temporal operators with first-order internalised formulae, mixing the explicit reference with the implicit reference of time.

In the combined approach [Gabbay 1991b], every temporalised formula α is associated with a first-order atomic formula $holds(t, \alpha)$, where α is now treated as a first-order term, and the free variables of α are considered free in $holds(t, \alpha)$. A set of axioms is added to combine the $holds(t, \alpha)$ formulae with the first-order internalised formulae, for example:

$$
\begin{array}{lcl}
holds(t, \alpha) & \leftrightarrow & (\alpha)^*[t], \text{ for all monolithic } \alpha \in \mathcal{L}_{\mathsf{L}} \\
holds(t, \alpha \wedge \beta) & \leftrightarrow & holds(t, \alpha) \wedge holds(t, \beta) \\
holds(t, S(\alpha, \beta)) & \leftrightarrow & \exists s[s < t \wedge holds(s, \alpha) \wedge \forall u(s < u < t \rightarrow holds(u, \beta))]
\end{array}
$$

etc.

As in the internalised approach, in the combined approach we still have to provide axioms for the flow of time.

Conclusions

We have shown in this chapter a way of composing an upper-level temporal logic system with a generic underlying logic system L and the resulting logic system $\mathsf{T(L)}$ was called the temporalisation of system L. We used the correspondence mapping method to prove soundness, completeness, decidability, conservativeness and separation for the temporalised logic system over linear flows of time. All those properties were initially properties of the temporal logic system. Many other properties remain to be analysed, such as compactness, finite model property and interpolation among others; the properties discussed here over classes of linear flows of time remain to be expanded for all classes of flows of time.

We need by no means restrict the upper-level logic system to temporal logic. In fact, the temporalisation presented here can be generalised to any propositional modal logic system M in the role of the upper-level logic system, so as to create a modalised logic system $\mathsf{M(L)}$. Its language and inference system can be obtained following the method we used to derive the those of $\mathsf{T(L)}$, based on the monolithic formulae of L. If the logic M has a possible world semantics, each possible world may be substituted by a model of L, so as to construct a model for the system $\mathsf{M(L)}$ in the same way a model was built for $\mathsf{T(L)}$. The correspondence mapping method may then be used to study how the properties of the modal logic system M are preserved in the modalised logic system $\mathsf{M(L)}$.

Chapter 3

Combinations of One-Dimensional Temporal Logics

We have seen in the previous chapter how to add a temporal dimension to a logic system. In particular, if a temporal logic is itself temporalised we obtain a two-dimensional temporal logic. Such a logic system is, however, very weakly expressive; if T is the internal (horizontal) temporal logic in the temporalisation process ($F \in \mathsf{T}$), and $\overline{\mathsf{T}}$ is the external (vertical) one ($\overline{F} \in \overline{\mathsf{T}}$), we cannot express that vertical and horizontal future operators commute,

$$F\overline{F}A \leftrightarrow \overline{F}FA.$$

In fact, the subformula $F\overline{F}A$ is not even in the temporalised language of $\overline{\mathsf{T}}(\mathsf{T})$, nor is the whole formula. In this chapter we study other methods of combining two *one-dimensional temporal logics* (1DTLs), into a *two-dimensional temporal logic* (2DTL), that are more expressive than the simple temporalisation of a 1DTL. Our approach to the study of such combinations of logics continues to be based on the analysis of how logical properties are transferred from the component logic systems to the combined, two-dimensional one. In this chapter we concentrate on the properties of soundness, completeness and decidability, and we study the methods for combining 1DTLs by *independent combination*, *full interlacing* and *restricted interlacing* into several different 2DTLs. Furthermore, we analyse the notion of a diagonal in a two-dimensional model.

Notation: Throughout this presentation, we refer to one of the temporal dimensions as the *horizontal dimension* and the other one as the *vertical dimension*; the symbols related to the vertical dimension are normally obtained by putting a bar on top of the corresponding horizontal ones, *e.g.* T and $\overline{\mathsf{T}}$, $F$and $\overline{F}$, $<$ and $\overline{<}$.

3.1 Introduction

In this chapter we introduce several methods for combining two one-dimensional temporal logics into a two-dimensional one. There are two distinct criteria for defining a modal/temporal logic system as two-dimensional:

(i) If the alphabet of the language contains two non-empty, disjoint sets of corresponding modal or temporal operators, Φ and $\overline{\Phi}$, each set associated to a distinct flow of time, $(T, <)$ and $(\overline{T}, \overline{<})$, then the system is two-dimensional.

(ii) If the truth value of a formula is evaluated with respect to two time points, then the system is two-dimensional. In this case, we even have the distinction between strong and weak interpretation of formulae that, as a consequence, generates different notions of valid formulae (a formulae is valid if it holds in all models for all pair of time points). Under the *strong interpretation*, the truth value of atoms depends on both dimensions, giving origin to *strongly valid formulae* when the evaluation of formulae is inductively extended to all connectives. In the *weak interpretation*, the truth value of atoms depends only on the one dimension, *e.g.* the horizontal dimension, giving origin to *weakly valid formulae*. Usually for this notion of two-dimensionality, both time points refer to the same flow of time, so we may also have the notion of (weak/strong) *diagonally valid* formulae by restricting validity to the case where both dimensions refer to the same point, *i.e.* A is diagonally valid iff $\mathcal{M}, t, t \models A$ for all $\mathcal{M}$ and t; see [Gabbay, Hodkinson and Reynolds 1994] for more details.

Criterion (i) above will sometimes be called the *syntactic criterion* for two-dimensionality, although it is not completely syntactic, *i.e.* it depends on the semantic notion of flows of time; criterion (ii) will be called the *semantic criterion* for two- dimensionality.

Note that both cases can yield, as an extreme case, one-dimensional temporal logic. In (i), by making $T = \overline{T}$ and $\overline{<} = (<)^{-1} = (>)$, *i.e.* by taking two flows with the same set of time points such that one order is the inverse of the other; in this case, the future operators $\Phi = \{F, G, U\}$ are associated with $(T, <)$ and the past operators $\overline{\Phi} = \{P, H, S\}$ are associated with $(T, >)$. In (ii), by fixing one dimension to a single time point so that the second dimension becomes redundant.

These two distinct approaches to the two-dimensionality of a system are independent. In fact, we will see in Section 3.2 a system that contains two distinct sets of operators over two classes of flows of time, but its formulae are evaluated at a single

point. On the other hand, there are several temporal logics in the literature satisfying (ii) but not (i), containing a single set of temporal operators in which formulae are evaluated according to two or more time points in the same flow [Aqvist 1979; Kamp 1971; Gabbay, Hodkinson and Reynolds 1994].

A logic system that respects both the syntactic and the semantic criteria for two-dimensionality is called *broadly two-dimensional*, and this will be the kind of system we will be aiming to achieve through combination methods; we consider in this work only strong evaluation and validity; the weak interpretation generates systems with the expressivity of only monadic first-order language [Gabbay, Hodkinson and Reynolds 1994], but for broadly two-dimensional systems we are interested in the expressivity of dyadic first-order language, although it is known that no set of temporal operators can be expressively complete[1] over dyadic first-order language [Venema 1990]. Venema's [1990] two-dimensional temporal logic, Segerberg's [1973] two-dimensional modal logic and the temporalisation of a temporal logic are all broadly two-dimensional; so are the combined logics in Sections 3.3 and 3.4.

In the study of one-dimensional temporal logics (1DTLs) several classes of flows of time are taken into account. When we move to 2DTLs, the number of such classes increases considerably, and every pair of one-dimensional classes can be seen as generating a different two-dimensional class. The study of 2DTLs would benefit much if the properties known to hold for 1DTLs could be systematically transferred to 2DTLs, avoiding the repetition of much of the work that has been published in the literature. This is a strong motivation to consider methods of combination of 1DTLs into 2DTLs and studying the transference of logical properties through each method. Also in favour of such an approach is the fact that the results concerning 2DTLs are then presented in a general, compact and elegant form.

In providing a method to combine two 1DTLs $\overline{\mathsf{T}}$ and T we have to pay attention to the following points:

(a) A method for combining logics $\overline{\mathsf{T}}$ and T is composed of three submethods, namely a method for combining the languages of $\overline{\mathsf{T}}$ and T, a method for combining their inference systems and a method for combining their semantics.

[1] A modal/temporal language is *expressively complete* over a class of first-order formulae if, for any first-order formula A in that class, there exists a modal/temporal formula B such that A is first-order equivalent to B^*, where B^* is the standard first-order translation of B [Gabbay, Hodkinson and Reynolds 1994].

(b) We study the combined logic system with respect to the way certain logical properties of $\overline{\mathsf{T}}$ and T are transferred to the two-dimensional combination. We focus here on the properties of soundness, completeness and decidability of the combined system given those of the component ones.

(c) The combined language should be able to express some properties of the interaction between the two-dimensions; otherwise the combination is just a partial one, and the two systems are not fully combined. For example, it is desirable to express formulae like $F\overline{F}A \leftrightarrow \overline{F}FA$ and $P\overline{F}A \leftrightarrow \overline{F}PA$ that are not in the temporalised language of $\overline{\mathsf{T}}(\mathsf{T})$.

(d) If we want to strengthen the interaction between the two systems, some properties of the interaction between the two-dimensions are expected to be theorems of the combining system, *e.g.* the commutativity of horizontal and vertical future operators such as $F\overline{F}A \leftrightarrow \overline{F}FA$ and $P\overline{F}A \leftrightarrow \overline{F}PA$.

(e) We want the combination method to be as independent as possible from the underlying flows of time.

All methods of combination must comply with item (a). The method for combining the languages of $\overline{\mathsf{T}}$ and T includes the choice of which sublanguage of $\overline{\mathsf{T}}$ and T is going to be part of the combined two-dimensional language, as well as the way in which this combination is done; in this presentation we will work, in the most general case, with the standard languages of S and U, $\overline{S}$ and $\overline{U}$, but we also consider some sublanguages, *e.g.* the monolithic subformulae in the case of temporalisation, or the sublanguage generated by a set of derived operators, *e.g.* the vertical "previous" ($\overline{\bullet}$) and "next" ($\overline{\bigcirc}$) in Section 3.4. In combining the inference systems of $\overline{\mathsf{T}}$ and T, we will assume that they are both an extension of classical logic and that they are presented in the form of a regular, normal axiomatic system $(\Sigma, \mathcal{I})$, where Σ is a set of axioms and $\mathcal{I}$ is a set of inference rules; one important requirement is that the combined system be a conservative extension of the two components. The combined semantics has to deal with the structure of the combined model, the evaluation of two-dimensional formulae over those structures and also with the combinations of classes of flows of time, *e.g.* in the temporalisation of $\mathsf{T}/\mathcal{K}$ with external $\overline{\mathsf{T}}/\overline{\mathcal{K}}$ we generated the logic $\overline{\mathsf{T}}(\mathsf{T})$ that is complete over the class $\mathcal{K}(\overline{\mathcal{K}})$.

Items (b), (c), (d) and (e) may conflict with each other. In fact, the rest of this chapter shows that this is the case, as we try to compromise between expressivity, independence of the underlying flow of time and the transference of logical properties.

3.2 Independent Combination

We have seen that the temporalisation process when applied to a temporal logic generates a weakly expressive 2DTL due to the syntactic restrictions imposed by the temporalisation method itself. The idea is then to define a new method of combination of logic systems that puts together all the expressivity of the two component logic systems; for that we assume that the language of a system is given by a set of formation rules. For example, we repeat here the formation rules of the language of US-temporal logic over a set of propositional atoms $\mathcal{P}$:

- every atomic proposition is in it;
- if A, B are in it, so are $\neg A$ and $A \wedge B$;
- if A, B are in it, so are $U(A, B)$ and $S(A, B)$.

Definition 3.1 Let $Op(\mathsf{L})$ be the set of non-boolean operators of a generic logic L. Let $\overline{\mathsf{T}}$ and T be logic systems such that $Op(\mathsf{T}) \cap Op(\overline{\mathsf{T}}) = \varnothing$. The *fully combined language* of logic systems $\overline{\mathsf{T}}$ and T over the set of atomic propositions $\mathcal{P}$, is obtained by the union of the respective set of connectives and the union of the formation rules of the languages of both logic systems. □

Let the operators U and S be in the language of US and $\overline{U}$ and $\overline{S}$ be in that of $\bar{\mathsf{U}}\bar{\mathsf{S}}$. Note that the renaming of the temporal operator is done prior to the combination, so that the combined systems contains the set of boolean operators $\{\neg, \wedge\}$ coming from both components, plus the set of temporal operators $\{U, S, \overline{U}, \overline{S}\}$. The formation rules of the languages of US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$ have the first two rules in common, so their fully combined language over a set of atomic propositions $\mathcal{P}$ is given by

- every atomic proposition is in it;
- if A, B are in it, so are $\neg A$ and $A \wedge B$;
- if A, B are in it, so are $U(A, B)$ and $S(A, B)$.
- if A, B are in it, so are $\overline{U}(A, B)$ and $\overline{S}(A, B)$.

In general, we do not want any non-boolean operator to be shared between the two languages, for this may cause problems when combining their axiomatisations. For example[2], if a generic operator $\Box$ belongs to both temporal logic system such

[2]this example is due to Ian Hodkinson

that T contains axiom $q \leftrightarrow \Box q$ and system $\overline{\mathsf{T}}$ contains axiom $\neg q \leftrightarrow \Box q$, the union of their axiomatisations will result in an inconsistent systems even though each system might have been itself consistent. To avoid such a behaviour the restriction $Op(\mathsf{T}) \cap Op(\overline{\mathsf{T}}) = \varnothing$ was imposed on the fully combined language of $\overline{\mathsf{T}}$ and T. Not only are the two languages taken to be independent of each other, but the set of axioms of the two systems are supposed to be disjoint; so we call the following combination method the *independent combination* of two temporal logics.

Definition 3.2 Let US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$ be two US-temporal logic systems defined over the same set $\mathcal{P}$ of propositional atoms such that their languages are independent. The *independent combination* $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ is given by the following:

- The fully combined language of US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$.

- If $(\Sigma, \mathcal{I})$ is an axiomatisation for US and $(\overline{\Sigma}, \overline{\mathcal{I}})$ is an axiomatisation for $\bar{\mathsf{U}}\bar{\mathsf{S}}$, then $(\Sigma \cup \overline{\Sigma}, \mathcal{I} \cup \overline{\mathcal{I}})$ is an axiomatisation for $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$. Note that the set of axioms Σ and $\overline{\Sigma}$ are supposed to be disjoint, but not the inference rules.

- The class of independently combined flows of time is $\mathcal{K} \oplus \overline{\mathcal{K}}$ composed of biordered flows of the form $(\tilde{T}, <, \overline{<})$ where the connected components of $(\tilde{T}, <)$ are in $\mathcal{K}$ and the connected components of $(\tilde{T}, \overline{<})$ are in $\overline{\mathcal{K}}$, and $\tilde{T}$ is the (not necessarily disjoint) union of the sets of time points T and $\overline{T}$ that constitute each connected component; such a biordered flow of time has been discussed in [Kracht and Wolter 1991] for the case of the independent combination of two mono-modal systems.

  A model structure for $\mathsf{T} \oplus \overline{\mathsf{T}}$ over $\mathcal{K} \oplus \overline{\mathcal{K}}$ is a 4-tuple $(\tilde{T}, <, \overline{<}, g)$, where $(\tilde{T}, <, \overline{<}) \in \mathcal{K} \oplus \overline{\mathcal{K}}$ and g is an assignment function $g : \tilde{T} \to 2^{\mathcal{P}}$.

  The semantics of a formula A in a model $\mathcal{M} = (\tilde{T}, <, \overline{<}, g)$ is defined as the union of the rules defining the semantics of $\mathsf{US}/\mathcal{K}$ and $\bar{\mathsf{U}}\bar{\mathsf{S}}/\overline{\mathcal{K}}$. The expression $\mathcal{M}, t \models A$ reads that the formula A is true in the (combined) model $\mathcal{M}$ at the point $t \in \tilde{T}$. The semantics of formulae is given by induction in the standard way:
  $\mathcal{M}, t \models p$ iff $p \in g(t)$ and $p \in \mathcal{P}$.
  $\mathcal{M}, t \models \neg A$ iff it is not the case that $\mathcal{M}, t \models A$.
  $\mathcal{M}, t \models A \wedge B$ iff $\mathcal{M}, t \models A$ and $\mathcal{M}, t \models B$.

$\mathcal{M}, t \models S(A,B)$ iff there exists an $s \in \tilde{T}$ with $s < t$ and $\mathcal{M}, s \models A$
and for every $u \in \tilde{T}$, if $s < u < t$ then $\mathcal{M}, u \models B$.
$\mathcal{M}, t \models U(A,B)$ iff there exists an $s \in \tilde{T}$ with $t < s$ and $\mathcal{M}, s \models A$
and for every $u \in \tilde{T}$, if $t < u < s$ then $\mathcal{M}, u \models B$.
$\mathcal{M}, t \models \overline{S}(A,B)$ iff there exists an $s \in \tilde{T}$ with $s \overline{<} t$ and $\mathcal{M}, s \models A$
and for every $u \in \tilde{T}$, if $s \overline{<} u \overline{<} t$ then $\mathcal{M}, u \models B$.
$\mathcal{M}, t \models \overline{U}(A,B)$ iff there exists an $s \in \tilde{T}$ with $t \overline{<} s$ and $\mathcal{M}, s \models A$
and for every $u \in \tilde{T}$, if $t \overline{<} u \overline{<} s$ then $\mathcal{M}, u \models B$.

□

Note that, despite the combination of two flows of time, formulae are evaluated according to a single point. The independent combination generates a system that is two-dimensional according to the first criterion but fails the second one, so it is not broadly two-dimensional.

The following result is due to [Thomason 1980] and is more general than the independent combination of two US-logics.

Proposition 3.1 *With respect to the validity of formulae, the independent combination of two modal logics is a conservative extension of the original ones.*

Note that we have defined conservative extension in Chapter 2 in proof theoretical terms; completeness for the independently combined case will lead to the conservativeness with respect to theorems.

As usual, we will assume that $\mathcal{K}, \overline{\mathcal{K}} \subseteq \mathcal{K}_{lin}$, so $<$ and $\overline{<}$ are transitive, irreflexive and total orders; similarly, we assume that the axiomatisations are extensions of $\mathsf{US}/\mathcal{K}_{lin}$.

The temporalisation process will be used as an inductive step to prove the transference of soundness, completeness and decidability for $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ over $\mathcal{K} \oplus \overline{\mathcal{K}}$. Let us first consider the *degree of alternation* of a $(\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}})$-formula A for US, $dg(A)$, and $\bar{\mathsf{U}}\bar{\mathsf{S}}$, $\overline{dg}(A)$.

$dg(p) = 0$	$\overline{dg}(p) = 0$
$dg(\neg A) = dg(A)$	$\overline{dg}(\neg A) = \overline{dg}(A)$
$dg(A \wedge B) = max\{dg(A), dg(B)\}$	$\overline{dg}(A \wedge B) = max\{\overline{dg}(A), \overline{dg}(B)\}$
$dg(S(A,B)) = max\{dg(A), dg(B)\}$	$\overline{dg}(\overline{S}(A,B)) = max\{\overline{dg}(A), \overline{dg}(B)\}$
$dg(U(A,B)) = max\{dg(A), dg(B)\}$	$\overline{dg}(\overline{U}(A,B)) = max\{\overline{dg}(A), \overline{dg}(B)\}$
$dg(\overline{S}(A,B)) = 1 + max\{\overline{dg}(A), \overline{dg}(B)\}$	$\overline{dg}(S(A,B)) = 1 + max\{dg(A), dg(B)\}$
$dg(\overline{U}(A,B)) = 1 + max\{\overline{dg}(A), \overline{dg}(B)\}$	$\overline{dg}(U(A,B)) = 1 + max\{dg(A), dg(B)\}$

Any formula A of $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ can be seen as a formula of some finite number of alternating temporalisations of the form $\mathsf{US}(\bar{\mathsf{U}}\bar{\mathsf{S}}(\mathsf{US}(\ \ldots\ )))$; more precisely, A can be seen as a formula of $\mathsf{US}(\mathsf{L}_n)$, where $dg(A) = n$, $\mathsf{US}(\mathsf{L}_0) = \mathsf{US}$, $\bar{\mathsf{U}}\bar{\mathsf{S}}(\mathsf{L}_0) = \bar{\mathsf{U}}\bar{\mathsf{S}}$, and $\mathsf{L}_{n-2i} = \bar{\mathsf{U}}\bar{\mathsf{S}}(\mathsf{L}_{n-2i-1})$, $\mathsf{L}_{n-2i-1} = \mathsf{US}(\mathsf{L}_{n-2i-2})$, for $i = 0, 1, \ldots, \lceil \frac{n}{2} \rceil - 1$.

Lemma 3.1 *Let* US *and* $\bar{\mathsf{U}}\bar{\mathsf{S}}$ *be two complete logic systems. Then,* A *is a theorem of* $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ *iff it is a theorem of* $\mathsf{US}(\mathsf{L}_n)$*, where* $dg(A) = n$*.*

Proof If A is a theorem of $\mathsf{US}(\mathsf{L}_n)$, all the inferences in its deduction can be repeated in $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$, so it is a theorem of $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$.

Suppose A is a theorem of $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$; let $B_1, \ldots, B_m = A$ be a deduction of A in $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ and let $n' = max\{dg(B_i)\}$, $n' \geq n$. We claim that each B_i is a theorem of $\mathsf{US}(\mathsf{L}_{n'})$. In fact, by induction on m, if B_i is obtained in the deduction by substituting into an axiom, the same substitution can be done in $\mathsf{US}(\mathsf{L}_{n'})$; if B_i is obtained by Temporal Generalisation from B_j, $j < i$, then by the induction hypothesis, B_j is a theorem of $\mathsf{US}(\mathsf{L}_{n'})$ and so is B_i; if B_i is obtained by Modus Ponens from B_j and B_k, $j, k < i$, then by the induction hypothesis, B_j and B_k are theorems of $\mathsf{US}(\mathsf{L}_{n'})$ and so is B_i.

So A is a theorem of $\mathsf{US}(\mathsf{L}_{n'})$ and, since US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$ are two complete logic systems, by Theorem 2.4, each of the alternating temporalisations in $\mathsf{US}(\mathsf{L}_{n'})$ is a conservative extension of the underlying logic; it follows that A is a theorem of $\mathsf{US}(\mathsf{L}_n)$, as desired. □

The transference of soundness, completeness and decidability follows directly from this result.

Theorem 3.1 (Independent Combination) *Let* US *and* $\bar{\mathsf{U}}\bar{\mathsf{S}}$ *be two sound, complete and decidable logic systems over the classes* $\mathcal{K}$ *and* $\overline{\mathcal{K}}$*, respectively. Then their independent combination* $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ *is sound, complete and decidable over the class* $\mathcal{K} \oplus \overline{\mathcal{K}}$*.*

Proof Soundness follows immediately from the validity of axioms and inference rules. For completeness, suppose that A is a consistent formula in $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$; by Lemma 3.1, A is consistent in $\mathsf{US}(\mathsf{L}_n)$, so we construct a temporalised model for it, and we obtain a model $(\tilde{T}_1, <_1, g_1, o_1)$ over $\mathcal{K}(\overline{\mathcal{K}}(\mathcal{K}(\ldots)))$, where o_1 is the "current time" necessary for the successive temporalisations. We show now how it can be transformed into a model over $\mathcal{K} \oplus \overline{\mathcal{K}}$.

Without loss of generality, suppose that US is the outermost logic system in $\mathsf{US}(\bar{\mathsf{U}}\bar{\mathsf{S}}(\mathsf{US}(\ \ldots\ )))$, and let n be the number of alternations. The construction is

recursive, starting with the outermost logic. Let $i \leq n$ denote the step of the construction; if i is odd, it is a US-temporalisation, otherwise it is a $\bar{\mathsf{U}}\bar{\mathsf{S}}$-temporalisation. At every step i we construct the sets $\tilde{T}_{i+1}$, $<_{i+1}$ and $\overline{<}_{i+1}$ and the function g_{i+1}.

We start the construction of the model at step $i = 0$ with the temporalised model $(\tilde{T}_1, <_1, g_1, o_1)$ such that $(\tilde{T}_1, <_1) \in \mathcal{K}$, and we take $\overline{<}_1 = \varnothing$. At step $i < n$, consider the current set of time points $\tilde{T}_i$; according to the construction, each $t \in \tilde{T}_i$ is associated to:

- a temporalised model $g_i(t) = (\tilde{T}^t_{i+1}, <^t_{i+1}, g^t_{i+1}, o^t_{i+1}) \in \mathcal{K}$ and take $\overline{<}^t_{i+1} = \varnothing$, if i is even; or
- a temporalised model $g_i(t) = (\tilde{T}^t_{i+1}, \overline{<}^t_{i+1}, g_{i+1}, o^t_{i+1}) \in \overline{\mathcal{K}}$ and take $<^t_{i+1} = \varnothing$, if i is odd.

The point t is made identical to $o^t_{i+1} \in \tilde{T}^t_{i+1}$, so as to add the new model to the current structure; note that this preserves the satisfiability of all formulae at t. Let $\tilde{T}_{i+1}$ be the (possibly infinite) union of all $\tilde{T}^t_{i+1}$ for $t \in \tilde{T}_i$; similarly, $<_{i+1}$ and $\overline{<}_{i+1}$ are generated. And finally, for every $t \in \tilde{T}_{i+1}$, the function g_{i+1} is constructed as the union of all g^t_{i+1} for $t \in \tilde{T}_i$.

Repeating this construction n times, we obtain a combined model over $\mathcal{K} \oplus \overline{\mathcal{K}}$, $\mathcal{M} = (\tilde{T}_n, <_n, \overline{<}_n, g_n)$, such that for all $t \in \tilde{T}_n$, $g_n(t) \subseteq \mathcal{P}$. Since satisfiability of formulae is preserved at each step, it follows that $\mathcal{M}$ is a model for A, and completeness is proved.

For decidability, again by Lemma 3.1, we can recursively apply the decision procedure of $\mathsf{US}(\mathsf{L}_n)$ and $\bar{\mathsf{U}}\bar{\mathsf{S}}(L_{n-1})$, starting with $n = dg(A)$, thus obtaining a decision procedure for $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$. □

3.3 Full Interlacing

With respect to the generation of two-dimensional systems, the method of independent combination has two main drawbacks. First, it generates logic systems whose formulae are evaluated at one single time point, not generating a broadly two-dimensional logic. Second, since the method independently combines the two component logic systems, no interaction between the dimension is provided by it. As a consequence, although a formula like $F\overline{F}A \leftrightarrow \overline{F}FA$ is expressible in its language, it will not be valid, as can easily be verified, for it expresses an interplay between the dimensions. We therefore introduce the notion of a *two-dimensional plane model*.

Definition 3.3 Let $\mathcal{K}$ and $\overline{\mathcal{K}}$ be two classes of flow of time. A *two-dimensional plane model* over the *fully combined class* $\mathcal{K} \times \overline{\mathcal{K}}$ is a 5-tuple $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$, where $(T, <) \in \mathcal{K}$, $(\overline{T}, \overline{<}) \in \overline{\mathcal{K}}$ and $g : T \times \overline{T} \to 2^{\mathcal{P}}$ is a two-dimensional assignment. The semantics of the horizontal and vertical operators are independent of each other.

$\mathcal{M}, t, x \models S(A, B)$ iff there exists $s < t$ such that $\mathcal{M}, s, x \models A$ and for all u, $s < u < t$, $\mathcal{M}, u, x \models B$.

$\mathcal{M}, t, x \models \overline{S}(A, B)$ iff there exists $y \overline{<} x$ such that $\mathcal{M}, t, y \models A$ and for all z, $y \overline{<} z \overline{<} x$, $\mathcal{M}, t, z \models B$.

Similarly for U and $\overline{U}$, the semantics of atoms and boolean connectives remaining the standard one. A formula A is (strongly) valid over $\mathcal{K} \times \overline{\mathcal{K}}$ if for all models $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$, for all $t \in T$ and $x \in \overline{T}$ we have $\mathcal{M}, t, x \models A$. □

With respect to the expressivity of fully combined two-dimensional languages, Venema [1990] has shown that no finite set of two-dimensional temporal operators is expressively complete over the class of linear flows with respect to dyadic first-order logic — despite the fact that US-temporal logic is expressively complete with respect to monadic first-order logic over $\mathbb{N}$ and over $\mathbb{R}$, and that, with additional operators (the Stavi operators), we can get expressive completeness over $\mathbb{Q}$ and $\mathcal{K}_{lin}$ [Gabbay 1981b]. So expressive completeness is not transferred by full interlacing.

It is easy to verify that the following formulae expressing the commutativity of future and past operators between the two dimensions are valid formulae in two-dimensional plane models.

I1 $F\overline{F}A \leftrightarrow \overline{F}FA$
I2 $F\overline{P}A \leftrightarrow \overline{P}FA$
I3 $P\overline{F}A \leftrightarrow \overline{F}PA$
I4 $P\overline{P}A \leftrightarrow \overline{P}PA$

Therefore, if we want to satisfy both the syntactic and the semantic criteria for two-dimensionality, we may define the method of *full interlacing* containing the fully combined language of US and $\overline{\mathsf{U}}\overline{\mathsf{S}}$ and their fully combined class of models. The question is whether there is a method for combining their axiomatisations so as to generate a *fully interlaced axiomatisation* that transfers the properties of soundness, completeness and decidability. The answer, however, is no, not in general. In some cases we can obtain the transference of completeness, in some other cases the transference fails. To illustrate that, we consider completeness results over classes of the form $\mathcal{K} \times \mathcal{K}$

3.3.1 The Completeness of $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$

We start by defining some useful abbreviations. Let p be a propositional atom; define:

$$hor(p) = \Box(p \wedge \overline{H}\neg p \wedge \overline{G}\neg p)$$
$$ver(p) = \overline{\Box}(p \wedge H\neg p \wedge G\neg p)$$

It is clear that $hor(p)$ makes p true along the horizontal line and false elsewhere; similarly for $ver(p)$ with respect to the vertical.

The axiomatisation of $\mathsf{US} \times \bar{\mathsf{U}}\bar{\mathsf{S}}$ over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$ extends that of $\mathsf{US} \oplus \bar{\mathsf{U}}\bar{\mathsf{S}}$ over $\mathcal{K}_{lin} \oplus \mathcal{K}_{lin}$ by including the interlacing axioms **I1**–**I4** and the following inference rules:

IR1 if $\vdash hor(p) \rightarrow A$ and p does not occur in A, then $\vdash A$
IR2 if $\vdash ver(p) \rightarrow A$ and p does not occur in A, then $\vdash A$

IR1 and **IR2** are two-dimensional extensions of the irreflexivity rule (IRR) defined in [Gabbay 1981a] for the one-dimensional case : if $\vdash p \wedge H\neg p \rightarrow A$ and p does not occur in A, then $\vdash A$. The use of **IR1** and **IR2** allows us to define U, S, $\overline{U}$ and $\overline{S}$ in terms of F, P, $\overline{F}$ and $\overline{P}$. So let r be an atom:

(U) $\quad r \wedge H\neg r \rightarrow [U(p,q) \leftrightarrow F(p \wedge H(Pr \rightarrow q))]$
$(\overline{U})$ $\quad r \wedge \overline{H}\neg r \rightarrow [\overline{U}(p,q) \leftrightarrow \overline{F}(p \wedge \overline{H}(\overline{P}r \rightarrow q))]$
(S) $\quad r \wedge H\neg r \rightarrow [S(p,q) \leftrightarrow P(p \wedge G(F(r \wedge H\neg r) \rightarrow q))]$
$(\overline{S})$ $\quad r \wedge \overline{H}\neg r \rightarrow [\overline{S}(p,q) \leftrightarrow \overline{P}(p \wedge \overline{G}(\overline{F}(r \wedge \overline{H}\neg r) \rightarrow q))]$

Therefore, we can base the proof of completeness in terms of F, P, $\overline{F}$ and $\overline{P}$ only.

Lemma 3.2 *The inference rules* **IR1** *and* **IR2** *are valid over* $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$.

Proof Suppose that $hor(p) \rightarrow A$ is a valid formula and p does not occur in A. Let $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$ be a model over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$, and let $t \in T$, $x \in \overline{T}$, so $\mathcal{M}, t, x \models hor(p) \rightarrow A$. Consider $\mathcal{M}' = (T, <, \overline{T}, \overline{<}, g')$ such that, for every $t' \in T$, $x' \in \overline{T}$, $p \in g'(t', x')$ iff $x = x'$ and, for $q \in \mathcal{P} - \{p\}$, $q \in h'(t', x')$ iff $q \in h(t', x')$. Clearly, $\mathcal{M}', t, x \models hor(p)$ and $hor(p) \rightarrow A$ is valid, so $\mathcal{M}', t, x \models A$. Since h and h' agree on all atoms in A, $\mathcal{M}, t, x \models A$, so **IR1** is valid. Similarly we show that **IR2** is valid. $\Box$

It follows directly that the axiomatisation is sound over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$.

The proof of completeness does not use directly the completeness results of the component $\mathsf{US}/\mathcal{K}_{lin}$ logics as in the previous methods; we use instead the technique of building a model out of (a special class of) maximal consistent sets. This construction applies techniques for the IRR rule described in [Gabbay and Hodkinson 1990]. The general construction strategy is the following. At every step we have a two-dimensional grid of special maximal consistent sets, IR*-theories. The use of **IR1** and **IR2** give us means to name every set in the grid with a pair of atoms that characterise the crossing of a vertical and a horizontal line, resembling a coordinate system. A step of the construction consists of finding a *counterexample* of a two-dimensional model in the current grid and fixing this counterexample by adding a new line to the grid, either vertical or horizontal, so as to end up with a new two-dimensional grid. The two-dimensional model is then obtained by taking the infinite union of all the grids.

IR*-theories

Let $\mathcal{P}$ be a set of proposition atoms and let $\mathcal{L}$ be the language generated by $\mathcal{P}$. A *maximal consistent set* (MCS) Γ over a language $\mathcal{L}$ is a consistent set of formulae such that there exists no consistent set Δ in $\mathcal{L}$ such that $\Gamma \subset \Delta$. An IR*-theory Γ over $\mathcal{P}$ is an MCS over $\mathcal{L}$ such that:

(a) there exists $u, v \in \mathcal{P}$ such that $hor(u) \wedge ver(v) \in \Gamma$;

(b) let a #-formula be one of the form $\#_0(B_0 \wedge \#_1(B_1 \wedge \ldots \wedge \#_m B_m)\ldots)$, where $\#_i \in \{F, P, \overline{F}, \overline{P}\}$; if $B \in \Gamma$ is a #-formula then there exists $u, v \in \mathcal{P}$ such that $\#_0(B_0 \wedge \#_1(B_1 \wedge \ldots \wedge \#_m[B_m \wedge hor(u) \wedge ver(v)])\ldots) \in \Gamma$.

The following motivates the definition of an order over IR*-theories.

Proposition 3.2 *For any Γ, Γ' MCSs, the following are equivalent:*

(a) whenever $A \in \Gamma$, we have $PA \in \Gamma'$,

(b) whenever $B \in \Gamma'$, we have $FB \in \Gamma$,

(c) whenever $GC \in \Gamma$, we have $C \in \Gamma'$,

(d) whenever $HD \in \Gamma'$, we have $D \in \Gamma$.

Similarly for the vertical operators.

For a proof with the horizontal operators, see Burgess [1984, lemma 1.6]. The proof for the vertical operators is analogous. Let $\mathcal{P}$ be a countably infinite set of atoms and let $\mathcal{S}$ be the set of all IR*-theories over $\mathcal{P}$. Define:

(a) $\Gamma \prec \Gamma'$ iff for all $GA \in \Gamma$, $A \in \Gamma'$.

(b) $\Gamma \overline{\prec} \Gamma'$ iff for all $\overline{G}A \in \Gamma$, $A \in \Gamma'$.

The axioms can be used in a standard way to show that $\prec$ and $\overline{\prec}$ define linear orders on $\mathcal{S}$. The following are straightforward generalisations of results in [Gabbay and Hodkinson 1990], the first of which uses directly **IR1** and **IR2**.

Proposition 3.3 *Let Γ be a consistent set of formulae over $\mathcal{P}_1$. Let $\mathcal{P}_2 \supseteq \mathcal{P}_1$ be an extension of $\mathcal{P}_1$ by a countably infinite set of atoms. Then there exists a IR*-theory $\Gamma' \supseteq \Gamma$ over $\mathcal{P}_2$.*

Proposition 3.4 *Let $\Gamma \in \mathcal{S}$. Then:*

(a) If $FA \in \Gamma$ then there exists $\Gamma' \in \mathcal{S}$ with $A \in \Gamma'$ and $\Gamma \prec \Gamma'$.

(b) If $PA \in \Gamma$ then there exists $\Gamma' \in \mathcal{S}$ with $A \in \Gamma'$ and $\Gamma' \prec \Gamma$.

(c) If $\overline{F}A \in \Gamma$ then there exists $\Gamma' \in \mathcal{S}$ with $A \in \Gamma'$ and $\Gamma \overline{\prec} \Gamma'$.

(d) If $\overline{P}A \in \Gamma$ then there exists $\Gamma' \in \mathcal{S}$ with $A \in \Gamma'$ and $\Gamma' \overline{\prec} \Gamma$.

The construction of the two-dimensional grid consists of basically of counterexample elimination and the filling of empty corners and gaps in the grid.

Definition 3.4 Let $\mathcal{G}$ be a set of IR*-theories such that $\Gamma_1, \Delta_1, \Delta_2 \in \mathcal{G}$. These sets form a $\llcorner$-*corner* in $\mathcal{G}$ if:

- $\Delta_1 \prec \Delta_2$;
- $\Delta_1 \overline{\prec} \Gamma_1$;
- there is no $\Psi \in \mathcal{G}$ such that $\Gamma_1 \prec \Psi$ and $\Delta_2 \overline{\prec} \Psi$.

Similarly, define $\urcorner$- $\lrcorner$- and $\ulcorner$-corners. □

We can "fill in" the corners due to the Corner Filling Lemma.

Lemma 3.3 *Let $\mathcal{G}$ be a set of IR*-theories such that $\Gamma_1, \Delta_1, \Delta_2 \in \mathcal{G}$ form a $\llcorner$-corner. Then there exists $\Psi \in \mathcal{S}$ such that $\Gamma_1 \prec \Psi$ and $\Delta_2 \overline{\prec} \Psi$.*

Proof Let $\Psi_0 = \{A \mid GA \in \Gamma_1\} \cup \{B \mid \overline{G}B \in \Delta_2\}$; we show it is consistent. Suppose not, then there is $A \in G\Gamma_1$ and $\overline{G}B \in \Delta_2$ such that $\vdash \neg(A \wedge B)$; then $F\overline{G}B \in \Delta_1$ and $\overline{P}F\overline{G}B \in \Gamma_1$; by axiom **I3**, $F\overline{PG}B \in \Gamma_1$, so $FB \in \Gamma_1$ and $F(A \wedge B) \in \Gamma_1$ which is a contradiction. So Ψ_0 is consistent.

So we extend Ψ_0 to an IR*-theory Ψ and clearly we will have $\Gamma_1 \prec \Psi$ and $\Delta_2 \overline{\prec} \Psi$. First, since $\Gamma_1, \Delta_2 \in \mathcal{S}$, there are atoms $u, v \in \mathcal{P}$ such that $hor(u) \in \Gamma_1$ and $ver(v) \in \Delta_2$ and $hor(u), ver(v) \in \Psi_0$.

Consider an enumeration $B_0, B_1, \ldots$ of two-dimensional formulae. Define Ψ_i by induction. Ψ_0 is defined and if Ψ_i is defined, set $\Psi_{i+1} = \Psi_i$ if $\Psi_i \cup \{B_i\}$ is inconsistent; otherwise, if B_i is not of a #-formula, set $\Psi_{i+1} = \Psi_i \cup \{B_i\}$.

Suppose B_i is a #-formula of the form $\#_0(C_0 \wedge \#_1(C_1 \ldots \wedge \#_m C_m) \ldots)$, where C_m is not a #-formula. Let $D = \bigwedge(\Gamma_i - \Gamma_0)$; this is a well defined formula since $\Gamma_i - \Gamma_0$ is finite. Then we must have $F(D \wedge B_i \wedge ver(v)) \in \Gamma_1$. For if $G\neg(D \wedge B_i \wedge ver(v)) \in \Gamma_1$, then $\neg(D \wedge B_i \wedge ver(v)) \in \Psi_0 \subseteq \Psi_i$, contradicting $\Psi_i \cup \{B_i\}$ is consistent. Now $F(D \wedge B_i)$ is itself a #-formula, so there are atoms $u_n, v_n \in \mathcal{P}$, $n = 0, \ldots, m$, such that, writing B_i' for $\#_0([C_0 \wedge hor(u_0) \wedge ver(v_0)] \wedge \ldots \wedge \#_m[C_m \wedge hor(u_m) \wedge ver(v_m)]) \ldots)$, we have that $F(D \wedge B_i' \wedge ver(v)) \in \Gamma_1$.

We claim that $\Psi_i \cup \{B_i'\}$ is consistent. Suppose not; then there is $GE \in \Gamma_1$ and $\overline{G}E' \in \Delta_2$ such that $\vdash \neg(D \wedge E \wedge E' \wedge B_i')$, so $G\neg(D \wedge E \wedge E' \wedge B_i') \in \Gamma_1$; since $ver(v) \in \Delta_2$, it follows that $\overline{G}(ver(v) \rightarrow E') \in \Delta_2$ and, by axiom **I3**, via Δ_1, we get $F(ver(v) \rightarrow E') \in \Gamma_1$, so $G(ver(v) \rightarrow E') \in \Gamma_1$; therefore $F(D \wedge B_i' \wedge ver(v) \wedge E) \in \Gamma_1$ and $F(D \wedge B_i' \wedge E' \wedge E) \in \Gamma_1$, which is a contradiction and proves the claim.

We then define $\Psi_{i+1} = \Psi_i \cup \{B_i, B_i'\}$ and set $\Psi = \bigcup_{i<\omega} \Psi_i$. That is clearly an IR*-theory, proving the result. □

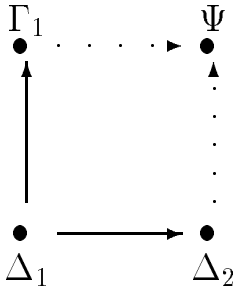


Figure 3.1 A ⌞-corner

The filling of a ⌞-corner is illustrated in Figure 3.1; similar versions of the Corner Filling Lemma are obtainable for the ⌝- ⌟- and ⌜-corners, but we omit the details. Not only corners have to be filled, but also gaps.

Definition 3.5 Let $\mathcal{G}$ be a set of IR*-theories such that $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2, \Phi \in \mathcal{G}$. These sets form a $\sqcup$-*gap* in $\mathcal{G}$ if:

- $\Gamma_1 \prec \Gamma_2$.
- $\Delta_1 \prec \Phi$, $\Phi \prec \Delta_2$;
- $\Delta_1 \overline{\prec} \Gamma_1$ and $\Delta_2 \overline{\prec} \Gamma_2$;
- there is no $\Psi \in \mathcal{G}$ such that $\Phi \overline{\prec} \Psi$ and $\Gamma_1 \prec \Psi \prec \Gamma_2$.

Similarly, define $\sqsubset$-$\sqsupset$- and $\sqcap$-gaps. □

The situation defined in a $\sqcup$-gap is illustrated in Figure 3.2. We then proceed to the Gap Filling Lemma.

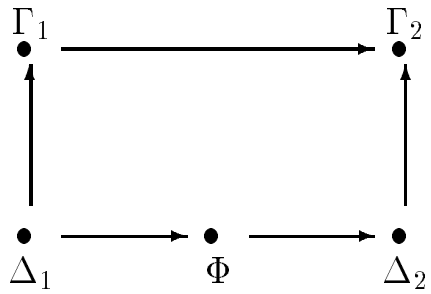


Figure 3.2 A $\sqcup$-gap

Lemma 3.4 *Let $\mathcal{G}$ be a set of IR*-theories such that $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2, \Phi \in \mathcal{G}$ form a $\sqcup$-gap. Then there exist $\Psi \in \mathcal{S}$ such that:*

(a) $\Phi \overline{\prec} \Psi$*; and*

(b) $\Gamma_1 \prec \Psi \prec \Gamma_2$.

Proof The Gap Filling Lemma gives us a $\Psi \in \mathcal{S}$ such that $\Phi \overline{\prec} \Psi$ and $\Gamma_1 \prec \Psi$. By linearity, we have that either $\Psi = \Gamma_2$ or $\Gamma_2 \prec \Psi$ or $\Psi \prec \Gamma_2$. Since $\Phi \in \mathcal{S}$, there is a $v \in \mathcal{P}$ such that $ver(v) \in \Phi$; it follows that $ver(v) \in \Psi$ and $Pver(v) \in \Delta_2$, so $\overline{P}Pver(v) \in \Gamma_2$ and, by axiom **I4**, $P\overline{P}ver(v) \in \Gamma_2$, so $P \in ver(v) \in \Gamma_2$, which contradicts both $\Psi = \Gamma_2$ and $\Gamma_2 \prec \Psi$, then $\Psi \prec \Gamma_2$. □

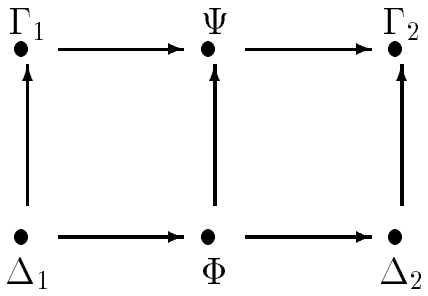


Figure 3.3 Filling a $\sqcup$-gap

Figure 3.3 illustrates the result of the previous Lemma as the "filling" of an incomplete$\sqcup$-gap. We can then define the two-dimensional grid.

The Two-dimensional Grid

A *two-dimensional grid* is a 5-tuple $\mathcal{G} = (X, <, \overline{X}, \overline{<}, f,)$ satisfying the following conditions.

C0 $X, \overline{X}$ are finite sets, $X, \overline{X} \subset \mathbb{Q}$ and $<, \overline{<}$ are the restriction of the standard order over $\mathbb{Q}$ to X and $\overline{X}$, respectively.

C1 f is a function from $f : X \times \overline{X} \to \mathcal{S}$ such that, for all $t, s \in X$ with $t < s$, and all $x, y \in \overline{X}$ with $x < y$:

$$f(t,x) \prec f(s,x) \quad \text{and}$$
$$f(t,x) \overline{\prec} f(t,y)$$

We write $\Gamma \in \mathcal{G}$ when we mean that Γ is an IR*-theory in the image of f.

We say that a grid $\mathcal{G}' = (X', <', \overline{X}', \overline{<}', f')$ is an *extension* of $\mathcal{G} = (X, <, \overline{X}, \overline{<}, f)$ iff $X \subseteq X'$, $< \subseteq <'$, $\overline{X} \subseteq \overline{X}'$, $\overline{<} \subseteq \overline{<}'$ and for all $t \in X$ and all $x \in \overline{X}$, $f(t,x) = f'(t,x)$.

A grid can also be decomposed in horizontal and vertical lines. The horizontal x-line in $\mathcal{G}$ is a 3-tuple $(X, <, f_x)$ such that, for every $t \in X$, $f_x(t) = f(t,x)$. Similarly, the vertical t-line in $\mathcal{G}$ is a 3-tuple $(\overline{X}, \overline{<}, \bar{f}_t)$ such that, for every $x \in \overline{X}$, $\bar{f}_t(x) = f(t,x)$. We write $(X, <, \tilde{f}_z)$ to denote a generic z-line, either horizontal or vertical. The extension of a z-line is defined in the obvious way.

Grid Initialisation

Let Γ_0 be a set of formulae consistent with the two-dimensional axiomatisation over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$. By Proposition 3.3, we extend it to a IR*-theory $\Gamma \supseteq \Gamma_0$ over a countably

infinite set of atoms $\mathcal{P}$. We then construct an initial grid $\mathcal{G}_0 = (X_0, <_0, \overline{X}_0, \overline{<}_0, f_0)$ by making:

$$X_0 = \overline{X}_0 = \{0\}$$
$$<_0 = \overline{<}_0 = \varnothing$$
$$f_0 = \{(0, 0, \Gamma)\}$$

which clearly satisfies **C0** and **C1**.

Counterexample Elimination

We say that (t, x, FA) is a *counterexample* for the grid $\mathcal{G} = (X, <, \overline{X}, \overline{<}, f)$ iff $PA \in f(t, x)$ and there is no $s \in X$ with $t < s$ such that $A \in f(s, x)$. Analogously, define the counterexamples (t, x, PA), $(t, x, \overline{F}A)$ and $(t, x, \overline{P}A)$.

Clearly, to every counterexample of the form (t, x, FA) and (t, x, PA) corresponds a one-dimensional counterexample in the horizontal x-line. And counterexamples of the form $(t, x, \overline{F}A)$ and $(t, x, \overline{P}A)$ find a corresponding one-dimensional counterexample in the vertical t-line.

Proposition 3.4 accomplishes the counterexample elimination for the one-dimensional cases. This is its two-dimensional counterpart.

Lemma 3.5 *Let $\mathcal{G} = (X, <, \overline{X}, \overline{<}, f)$ be a two-dimensional grid and let c be a counterexample to it. Then there exists an extension $\mathcal{G}' = (X', <', \overline{X}', \overline{<}', f')$, to which c is no longer a counterexample.*

Proof Initially, make $f \subseteq f'$. By symmetry, assume without loss of generality, that c is of the form (t, x, FA); for the other cases the proof is completely analogous. By Proposition 3.4, there is an IR*-theory Γ' such that $A \in \Gamma'$ and $f(t, x) \prec \Gamma'$; for any $s \in X$, $\Gamma' \neq f(s, x)$, otherwise c would not be a counterexample. Let t_{max} be the largest element of X and for every $t \in X$, $t < t_{max}$, let t' designate its immediate successor in X.

If $f(t_{max}, x) \prec \Gamma'$, then make $t^* = t_{max} + 1 \in \mathbb{Q}$. Let $X' = X \cup \{t^*\}$, $\overline{X}' = \overline{X}$ and $f'(t^*, x) = \Gamma'$, therefore generating a corner in $\mathcal{G} \cup \Gamma'$. By successive applications of the suitable version of the Corner Filling Lemma, $f'(t^*, y)$ can be determined for every $y \in \overline{X}'$, therefore adding a new vertical t^*-line to the grid.

Otherwise, $\Gamma' \prec f(t_{max}, x)$ and by linearity there exists $s \in X$, $t < s < t_{max}$, such that $f(s, x) \prec \Gamma' \prec f(s', x)$. Make $t^* = \frac{s+s'}{2} \in \mathbb{Q}$, $X' = X \cup \{t^*\}$, $\overline{X}' = \overline{X}$ and $f'(t^*, x) = \Gamma'$, therefore generating a gap in $\mathcal{G} \cup \Gamma'$. By successive applications of the suitable version of the Gap Filling Lemma, determine $f'(t^*, y)$, for every $y \in \overline{X}'$, thus adding a new vertical t^*-line to the grid.

We have thus extended the original grid into $\mathcal{G}' = (X', <', \overline{X}', \overline{<}', f')$ where c is no longer a counterexample by adding a new vertical line. Similarly for the other types of counterexamples where, if c is of the form $(t, x, \overline{F}A)$ or$(t, x, \overline{P}A)$, we add a new horizontal line. □

To construct the two-dimensional grid, we start with $\mathcal{G}_0$ and assume we have constructed $\mathcal{G}_n$. For every n, there are at most countably many counterexamples; suppose there is an order on formulae by which the counterexamples are ordered. Choose c as the first counterexample and, by application of Lemma 3.5, we obtain $\mathcal{G}_{n+1}$. Take $\mathcal{G}^* = (X^*, <^*, \overline{X}^*, \overline{<}^*, f^*) = \bigcup_{n<\omega} \mathcal{G}_n$. Clearly, there are no counterexamples left in $\mathcal{G}^*$ and $(X^*, <^*)$,$(\overline{X}^*, \overline{<}^*) \in \mathcal{K}_{lin}$ and $\mathcal{G}^*$ satisfies **C1** plus:

C2a For all $t \in X^*$ and $x \in \overline{X}^*$, $FA \in f^*(t, x)$ iff there exists $s \in X^*$, $t <^* s$, such that $A \in f^*(s, x)$.

C3a For all $t \in X^*$ and $x \in \overline{X}^*$, $\overline{F}A \in f^*(t, x)$ iff there exists $y \in \overline{X}^*$, $x \,\overline{<}^* y$, such that $A \in f^*(t, y)$.

and their mirror images, **C2b** and **C3b**. Since we are dealing with IR*-theories, axioms (U), $(\overline{U})$, (S) and $(\overline{S})$ make $\mathcal{G}^*$ satisfy:

C4a For all $t \in X^*$ and $x \in \overline{X}^*$, $U(A, B) \in f^*(t, x)$ iff there exists $s \in X^*$, $t <^* s$, such that $A \in f^*(s, x)$ and for all u, $t <^* u <^* s$, $B \in f^*(u, x)$.

C5a For all $t \in X^*$ and $x \in \overline{X}^*$, $\overline{U}(A, B) \in f^*(t, x)$ iff there exists $y \in \overline{X}^*$, $x \,\overline{<}^* y$, such that $A \in f^*(t, y)$ and for all z, $x \,\overline{<}^* z \,\overline{<}^* y$, $B \in f^*(t, z)$.

plus their mirror images (**C4b**, **C5b**) .

For every atom p define h as

$$p \in h(x, y) \text{ iff } p \in f^*(x, y)$$

Let $\mathcal{M} = (X^*, <^*, \overline{X}^*, \overline{<}^*, h)$ be a two-dimensional model; by a simple induction on A, it can be show that, for every A,

$$\mathcal{M}, t, x \models A \text{ iff } A \in f^*(t, x)$$

It follows that $\mathcal{M}, 0, 0 \models \Gamma_0$.

We have thus proved the following theorem.

Theorem 3.2 (2D-completeness) *There is a sound and complete axiomatisation over the class of full two-dimensional temporal models over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$.*

Furthermore, it is possible to obtain completeness results for other classes of two-dimensional plane flows based on this construction along the usual steps used in the one-dimensional case; the proof is in Appendix B, Theorem B.1.

Theorem 3.3 (2D-completeness) *There are sound and complete axiomatisations over the two-dimensional plane classes $\mathcal{K}_{dis}\times\mathcal{K}_{dis}$, $\mathbb{Q}\times\mathbb{Q}$, $\mathcal{K}_{lin}\times\mathcal{K}_{dis}$, $\mathcal{K}_{lin}\times\mathbb{Q}$ and $\mathcal{K}_{dis}\times\mathbb{Q}$.*

3.3.2 Incompleteness Results

The negative result is the following.

Proposition 3.5 (2D-unaxiomatisability) *There are no finite axiomatisations for the (strongly) valid two-dimensional formulae over the classes $\mathbb{Z}\times\mathbb{Z}$, $\mathbb{N}\times\mathbb{N}$ and $\mathbb{R}\times\mathbb{R}$.*

This proposition follows directly from Venema's proof that the valid formulae over the upper half two-dimensional plane are not enumerable for $\mathbb{Z}\times\mathbb{Z}$, $\mathbb{N}\times\mathbb{N}$ and $\mathbb{R}\times\mathbb{R}$, which in its turn was based on [Halpern and Shoham 1986]. Since there are sound, complete and decidable US-temporal logics over $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{R}$ [Reynolds 1992], the general conclusion on full interlacing is the following.

Theorem 3.4 (Full Interlacing) *Completeness and decidability do not transfer in general through full interlacing.*

It has to be noted that two-dimensional temporal logics seem to behave like modal logics in the following sense. We can see the result of the independent combination of US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$ as generating a "minimal" combination of the logics, *i.e.* one without any interference between the dimensions. The addition of extra axioms, inference rules or an extra condition on its models has to be studied on its own, just as adding a new axiom to a modal logic or imposing a new property on its accessibility relation has to be analysed on its own.

The full interlacing method illustrates the conflict between the generality of a method and its ability to achieve the transference of logical properties. We next restrict the interlacing method so as to recover the transference of logical properties.

3.4 Restricted Interlacing

The fact that the transference of logical properties fails for the interlacing of two US-temporal logics does not mean that the interlacing of any two temporal logic systems fails to achieve this transference. We restrict the vertical logic system to a temporal logic $\bar{\mathsf{N}}\bar{\mathsf{P}}$ with operators $\overline{\bigcirc}$ for Next time and $\overline{\bullet}$ for Previous time; the formation rules for the formulae of $\bar{\mathsf{N}}\bar{\mathsf{P}}$ are the standard ones. This is a restriction of the $\bar{\mathsf{U}}\bar{\mathsf{S}}$-language for $\overline{\bigcirc}$ and $\overline{\bullet}$ can be defined in terms of $\overline{U}$ and $\overline{S}$, namely by

$$\overline{\bigcirc}A =_{def} \overline{U}(A, \bot)$$
$$\overline{\bullet}A =_{def} \overline{S}(A, \bot)$$

Not only is the expressivity of the language reduced this way, but also the underlying flow of time is now restricted to a discrete one; in fact, we concentrate our attention on integer-like flows of time.

Let $h : \mathbb{Z} \to \mathcal{P}$ be a temporal assignment over the integers so that the semantics of $\bar{\mathsf{N}}\bar{\mathsf{P}}$ over the integers is the usual for atoms and boolean operators and

$$(\mathbb{Z}, <, h), t \models \overline{\bigcirc}A \quad \text{iff} \quad (\mathbb{Z}, <, h), t+1 \models A$$
$$(\mathbb{Z}, <, h), t \models \overline{\bullet}A \quad \text{iff} \quad (\mathbb{Z}, <, h), t-1 \models A$$

An axiomatisation for $\mathsf{NP}/\mathbb{Z}$ is given by the classical tautologies plus

NP1 $\overline{\bigcirc}\overline{\bullet}p \to p$
NP2 $\overline{\bigcirc}\neg p \leftrightarrow \neg\overline{\bigcirc}p$
NP3 $\overline{\bigcirc}(p \wedge q) \to \overline{\bigcirc}p \wedge \overline{\bigcirc}q$
NP4 The mirror image of **NP1–3** obtained by interchanging $\overline{\bigcirc}$ with $\overline{\bullet}$

The rules of inference are the usual Substitution, Modus Ponens and Temporal Generalisation (from A infer $\overline{\bigcirc}A$ and $\overline{\bullet}A$).

The converse of each axiom can be straightforwardly derived, so the formulae on both sides of the $\to$-connective are actually equivalent. It follows that every $\bar{\mathsf{N}}\bar{\mathsf{P}}$-formula can be transformed into an equivalent one by "pushing in" the temporal operators, e.g. by following the arrows the axioms, and by "cancelling" the occurrences of $\overline{\bigcirc}$ and $\overline{\bullet}$ in a string of temporal operators, e.g. $\overline{\bigcirc}\overline{\bullet}\overline{\bullet}\overline{\bigcirc}\overline{\bullet}p$ is equivalent to $\overline{\bullet}p$; the resulting $\bar{\mathsf{N}}\bar{\mathsf{P}}$-*normal form* formula is a boolean combination of formulae of the form $\overline{\bigcirc}^k p$ and $\overline{\bullet}^l q$, where p and q are atoms, $k, l \in \mathbb{N}$ and $\overline{\bigcirc}^k$ is a sequence of $\overline{\bigcirc}$-symbols of size k, similarly for $\overline{\bullet}^l$; it is useful sometimes to consider k negative or 0, so we define $\overline{\bigcirc}^{-k}A = \overline{\bullet}^k A$ and $\overline{\bigcirc}^0 A = A$. As an example, the formula $\overline{\bigcirc}\overline{\bigcirc}(\overline{\bullet}\overline{\bullet}\overline{\bullet}(p \wedge q) \vee p)$ has normal form $(\overline{\bullet}p \wedge \overline{\bullet}q) \vee \overline{\bigcirc}\overline{\bigcirc}p$. The existence of

such normal form gives us very simple proofs for completeness and decidability of $\bar{\mathsf{N}}\bar{\mathsf{P}}/\mathbb{Z}$ that we outline next.

For completeness, let Σ be a possibly infinite consistent set of $\bar{\mathsf{N}}\bar{\mathsf{P}}$-formulae and assume all formulae in the set is in the normal form. Σ can be seen as a consistent set of propositional formulae where each maximal subformulae of the form $\overline{\bigcirc}^k p$ is understood as a new propositional atom, so let h_0 be a propositional valuation assigning every extended atom into $\{true, false\}$. For $n \in \mathbb{Z}$, let $h(n) = \{p \in \mathcal{P} \mid h_0(\overline{\bigcirc}^n p) = true\}$. Clearly $(\mathbb{Z}, <, h)$ is a model for the original set.

For decidability, let A be a formula of $\bar{\mathsf{N}}\bar{\mathsf{P}}$ and let A^* be its normal form; clearly there exists an algorithm to transform A into A^*. By considering subformulae of the form $\overline{\bigcirc}^k p$ as new atoms, k possibly negative, we apply any decision procedure for propositional logic to A^*. A is a $\bar{\mathsf{N}}\bar{\mathsf{P}}$-valid formula iff A^* is a propositional tautology.

Definition 3.6 The *restricted interlacing* of temporal logic systems $\mathsf{US}/\mathcal{K}$ and $\bar{\mathsf{N}}\bar{\mathsf{P}}/\mathbb{Z}$ is the two-dimensional temporal logic system $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$ given by:

- the fully combined language of US and $\bar{\mathsf{N}}\bar{\mathsf{P}}$;
- the two-dimensional plane model over $\mathcal{K} \times \mathbb{Z}$, equipped with the broadly two-dimensional semantics;
- the union of the axioms of $\mathsf{US}/\mathcal{K}$ and $\bar{\mathsf{N}}\bar{\mathsf{P}}/\mathbb{Z}$ plus the interlacing axioms

  $$\overline{\bigcirc} U(p,q) \rightarrow U(\overline{\bigcirc} p, \overline{\bigcirc} q)$$
  $$\overline{\bigcirc} S(p,q) \rightarrow S(\overline{\bigcirc} p, \overline{\bigcirc} q)$$

  plus their duals obtained by swapping $\overline{\bigcirc}$ with $\overline{\bullet}$; the inference rules are just the union of the inference rules of both component systems. □

The following gives us a normal form for $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$.

Lemma 3.6 *Let A be a formula of $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$. There exists a normal form formula A^* equivalent to A, such that all the occurrences of $\overline{\bigcirc}$ and $\overline{\bullet}$ in it are in the form $\overline{\bigcirc}^k p$ and $\overline{\bullet}^l q$, where p and q are atoms.*

The proof is in Appendix B, Lemma B.1.

Theorem 3.5 (Completeness via restricted interlacing) *Let US be a logic system complete over the class $\mathcal{K} \subseteq \mathcal{K}_{lin}$. Then the two-dimensional system $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$ is complete over $\mathcal{K} \times \mathbb{Z}$.*

Proof Consider a $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$-consistent formula A and assume it is in the normal form. So we can see A as a US-formulae over the extended set of atoms $\overline{\bigcirc}^k$, k possibly negative or 0. From the completeness of $\mathsf{US}/\mathcal{K}$ there exist a one-dimensional model $(T, <, h_{US})$ for A at a point $o \in T$, where $(T, <) \in \mathcal{K}$. Define the two-dimensional assignment

$$h(k, t) = \{p \in \mathcal{P} \mid \overline{\bigcirc}^k p \in h_{US}(t)\}.$$

Clearly, $(T, <, \mathbb{Z}, <_{\mathbb{Z}}, h)$ is a two-dimensional plane $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$-model for A at $(o, 0)$. □

Corollary 3.1 *If* $\mathsf{US}/\mathcal{K}$ *is strongly complete, so is* $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}/\mathcal{K} \times \mathbb{Z}$.

Theorem 3.6 (Decidability via restricted interlacing) *If the logic system* US *is decidable over* $\mathcal{K}$*, so is* $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$ *over* $\mathcal{K} \times \mathbb{Z}$.

Proof The argument of the proof is the same as that of the decidability of NP, all we have to do is note that there exists an algorithmic way to convert a combined two-dimensional formula into its normal form, so it can be seen as a US-formula and we can apply the US-decision procedure to it. □

So by restricting the expressivity and the underlying class of flows of time, we can obtain the transference of the basic logical properties via restricted interlacing. It should not be difficult to extend these results to $\mathbb{N}$ instead of $\mathbb{Z}$, although we do not explore this possibility here.

3.5 The Two-dimensional Diagonal

We now study some properties of the diagonal in two-dimensional plane models. The diagonal is a privileged line in the two-dimensional model intended to represent the sequence of time points we call "now", *i.e.* the time points on which an historical observer is expected to be traverse . The observer is, therefore, on the diagonal when he or she poses a query (*i.e.* evaluates the truth value of a formula) on a two-dimensional model. The diagonal is illustrated in Figure 3.4

So let δ be a special atom and consider the formulae:

D1 $\Diamond\delta \wedge \overline{\Diamond}\delta$

D2 $\delta \rightarrow (G\neg\delta \wedge H\neg\delta \wedge \overline{G}\neg\delta \wedge \overline{H}\neg\delta)$

D3 $\delta \rightarrow (\overline{H}G\neg\delta \wedge \overline{G}H\neg\delta)$

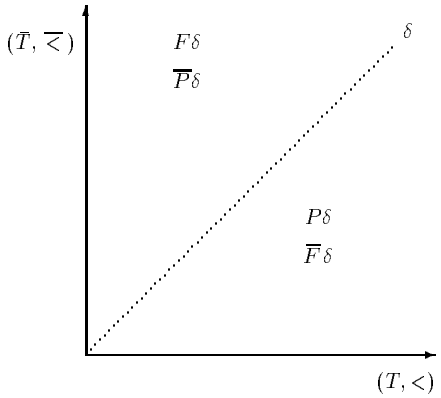


Figure 3.4 The two-dimensional diagonal

Let $Diag = \Box\overline{\Box}(\mathbf{D1} \wedge \mathbf{D2} \wedge \mathbf{D3})$. The intuition behind $Diag$ is the following. **D1** implies that the two-dimensional diagonal can always be reached in both vertical and horizontal directions; **D2** implies that there are no two diagonal points on the same horizontal line and on the same vertical line and **D3** implies that the diagonal goes in the direction SW–NE. We say that $Diag$ characterises a two-dimensional diagonal in the following sense.

Lemma 3.7 *Let $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$ be a full two-dimensional model over $\mathcal{K} \times \overline{\mathcal{K}}$, $\mathcal{K}, \overline{\mathcal{K}} \subseteq \mathcal{K}_{lin}$, and let δ be a propositional letter. Then the following are equivalent.*

(a) $\mathcal{M}, t, x \models Diag$, for some $t \in T$ and $x \in \overline{T}$.

(b) $\mathcal{M}, t, x \models Diag$, for all $t \in T$ and $x \in \overline{T}$.

(c) There exists an isomorphism $i : T \to \overline{T}$ such that $\mathcal{M}, t, x \models \delta$ iff $x = i(t)$.

Proof It is straightforward to show that (a) $\iff$ (b) and (c) $\implies$ (a); we show only (b) $\implies$ (c). So assume that $\mathcal{M}, t, x \models Diag$, for all $t \in T$ and $x \in \overline{T}$. Define

$$i = \{(t, x) \in T \times \overline{T} \mid \mathcal{M}, t, x \models \delta\}.$$

All we have to show is that i is an isomorphism.

- i, i^{-1} are functions such that $dom(i) = T$ and $dom(i^{-1}) = \overline{T}$. Suppose $(t, x_1), (t, x_2) \in i$; then $\mathcal{M}, t, x_1 \models \delta$ and $\mathcal{M}, t, x_2 \models \delta$. By linearity of $\overline{T}$, $x_1 = x_2$, $x_1 \overline{<} x_2$ or $x_2 \overline{<} x_1$, but **D2** eliminates the latter two; **D1** gives us that $dom(i) = T$. Similarly, the linearity of T and **D2** gives us that i^{-1} is a function and **D1** gives us that $dom(i^{-1}) = \overline{T}$.

- $i(t) = x$ iff $i^{-1}(x) = t$ follows directly from the definition. So i is a bijection.

- i preserves ordering. Suppose $t_1 < t_2$; by the linearity of $\overline{T}$ we have three possibilities:
  - $i(t_1) = i(t_2)$ contradicts i is a bijection.
  - $i(t_2) \overline{<} i(t_1)$ contradicts **D3**.
  - $i(t_1) \overline{<} i(t_2)$ is the only possible option.

Therefore i is an isomorphism, which proves the result. □

This result shows that by adding **D1**–**D3** to the axiomatisation over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$ of Section 3.3 gives us completeness over the class of models of the form $(T, <, T, <, g)$, $(T, <) \in \mathcal{K}_{lin}$. It follows from [Halpern and Shoham 1986], however, that such logic system is undecidable.

The diagonal is interpreted as the sequence of time points we call "now". The diagonal divides the two-dimensional plane in two semi-planes. The semi-plane that is to the (horizontal) left of the diagonal is "the past", and the formula $F\delta$ holds over all points of this semi- plane. Similarly, the semi-plane that is to the (horizontal) right of the diagonal is "the future", and the formula $P\delta$ holds over all points of this semi-plane. Figure 3.4 puts this fact in evidence. If we assume that $Diag$ holds over $\mathcal{M}$ such that i is the isomorphism defined in Lemma 3.7, $t < s$ iff $i(t) \overline{<} i(s)$, then

$$\begin{aligned} \mathcal{M}, t, x \models F\delta &\text{ iff exists } s > t \text{ such that } \mathcal{M}, s, x \models \delta \text{ and } i(s) = x \\ &\text{iff exists } y = i(t) \overline{<} x \text{ such that } \mathcal{M}, t, y \models \delta \\ &\text{iff } \mathcal{M}, t, x \models \overline{P}\delta. \end{aligned}$$

Similarly, it can be shown that:

$$\mathcal{M}, t, x \models P\delta \text{ iff } \mathcal{M}, t, x \models \overline{F}\delta.$$

It follows that the following formula is valid for $\mathsf{US} \times \bar{\mathsf{U}}\bar{\mathsf{S}}$ over $\mathcal{K}_{lin} \times \mathcal{K}_{lin}$:

$$Diag \rightarrow (\ (F\delta \leftrightarrow \overline{P}\delta) \ \wedge \ (P\delta \leftrightarrow \overline{F}\delta) \ ).$$

As a consequence, $\overline{P}\delta$ holds over all points of the "past" semi-plane and $\overline{F}\delta$ holds over all points of the "future" semi-plane, as is indicated in Figure 3.4.

It would be desirable to generalise the idea of a diagonal as the sequence of "now" moments to any pair of flows of time that are not necessarily isomorphic. For that, we would have to create an order between the points of the two flows, *i.e.* we would have to merge the flows.

So let $(T, <)$ and $(\overline{T}, \overline{<})$ be two flows of time such that T and $\overline{T}$ are disjoint. Then there always exists a flow $(T', <')$ and a mapping $f : T \cup \overline{T} \rightarrow T'$ such that f

is one-to-one and order preserving. The *f-merge of* $(T, <)$ *and* $(\overline{T}, \overline{<})$ is the flow of time consisting of the image of f ordered by the restriction of $<'$ to the image of f. An example of an f-merge is shown in Figure 3.5, where $f(y)$ is made equal, via merge, to $f(\bar{x})$ and on the merged flow the order is preserved, *i.e.* originally $x < y$ and $\bar{x} \overline{<} \bar{y}$ and on the f-merged flow $f(x) <' f(y) = f(\bar{x}) <' f(\bar{y})$.

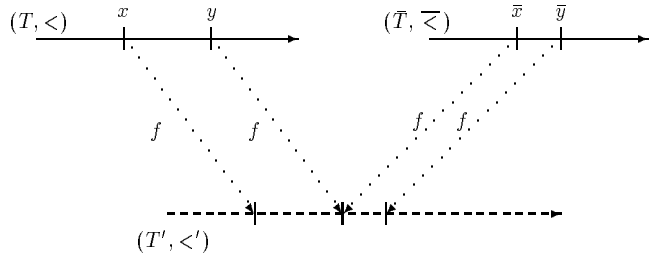


Figure 3.5 The f-merge

We can then construct a two dimensional model with two copies of the f-merge, in which we can define a diagonal over $(T', <') \times (T', <')$ as shown in Figure 3.6.

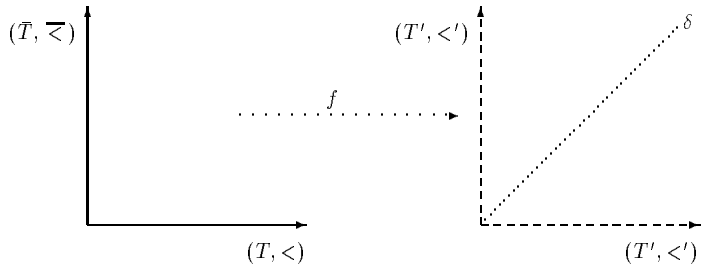


Figure 3.6 The diagonal of two distinct flows

This construction motivates a method of combining two one-dimensional temporal logics into another one-dimensional logic, namely that over the class of all f-merges of its two-component flows of time. We could then study the transference of logical properties in the same way as we have done in this and the previous chapter, but we do not investigate those matters here.

Chapter 4

Temporal Database Updates

In this chapter we set the temporal database framework in which the past, the present and even the future can be changed. For that, we follow [Chomicki and Niwiński 1993] by defining the notion of a temporal database as a model of first-order temporal logic (FOTL); the notion of a temporal query is defined on such a framework.

That framework, however, contrasts with the presentation so far of propositional temporal logics. To bring us back to a propositional framework, we consider the propositional abstraction of temporal databases, which then allows us to apply the full propositional two-dimensional model for the description of temporal database evolution and helps us to characterise the class of acceptable temporal database updates.

The two-dimensional description of temporal database evolution presented here restricts the generic view of combinations of linear temporal logics, concentrating basically on discrete flows of time.

We compare the two temporal database notions of valid-time database and transaction-time database in the light of their two-dimensional evolution, providing a formal characterisation of the differences between those two kinds of temporal databases.

4.1 A Logical View of Temporal Databases

We give here a formal presentation of temporal databases in a temporal logic perspective. For that, we start by analysing the language and semantics of first-order temporal logic, FOTL, over a finite signature. Finiteness is an important property to be taken into account in the database presentation, for databases are supposed

to be finite repositories of information. That will motivate us to define temporal queries as the restricted class of *safe* FOTL-formulae.

The Language

We consider a first-order language, without function symbols but including equality, fully combined with a US-temporal logic over the integers; throughout this chapter and the next one, we restrict ourselves to discrete, integer-like flows of time.

A *database signature* is a pair $\mathcal{S} = (\mathcal{S}_C, \mathcal{S}_P)$, where $\mathcal{S}_C$ is a countable set of constant symbols and $\mathcal{S}_P$ is a set of predicate symbols, such that each predicate symbol is associated with an arity $r \geq 1$; the signature is finite if $\mathcal{S}_P$ is finite. Let V be a countably infinite set of variables; a *term* is either a constant or a variable (functional symbols are normally absent from databases); we consider the existential quantifier, $\exists$, as primitive and syntactically define the universal quantifier as $\forall \equiv \neg\exists\neg$; for reasons that will become clear when we discuss the safeness of queries, we will consider both $\wedge$ and $\vee$ boolean connectives as primitive. The *atomic formulae* of FOTL are of the form $a_1 = a_2$ and $p(a_1, \ldots, a_r)$, where a_i's are terms and p is a predicate symbol of arity r. The set of free variables of a formula A is represented by $free(A)$. The language of FOTL over the signature $(\mathcal{S}_C, S_P)$ is defined as:

- every atomic formula is in it, the set of variables occurring in an atomic formula are all free;
- if A, B are in it, so are $\neg A$, $A \wedge B$, and $A \vee B$, with free variables, respectively, $free(A)$, $free(A) \cup free(B)$ and $free(A) \cup free(B)$;
- if A, B are in it, so are $U(A, B)$ and $S(A, B)$, with free variables $free(A) \cup free(B)$;
- if A is in it, so is $\exists x A$, with free variables $free(A) - \{x\}$.

We write $A(x_1, \ldots, x_m)$ to indicate that $x_1, \ldots, x_m$ are all the free variables of A. A formula is *ground* if it contains no variables nor quantifiers. We consider also the usual syntactical definitions of other temporal operators as $\bigcirc$, F and H in terms of S and U. We also say that a subformula (or, for that matter, any symbol) occurs positively in a formula if it occurs within the scope of an even number of $\neg$ symbols; it occurs negatively if it occurs within the scope of an odd number of $\neg$ symbols.

Example 4.1 Consider the following finite signature $\mathcal{S} = (\mathcal{S}_C, \mathcal{S}_P)$:

$\mathcal{S}_C = \{\text{strings of characters}\}$
$\mathcal{S}_P = \{employee\}$

where $employee(Name, Salary, Department)$ is a three place predicate symbol. The following are well formed formulae over this signature:

$employee(\text{Peter}, \text{2K}, \text{Marketing})$;
$\bullet employee(x, \text{2K}, \text{Marketing}) \wedge \neg(x = \text{Peter})$;
$P \exists y \exists z\, employee(x, y, z) \wedge \neg \exists y_1 \exists z_1\, employee(x, y_1, z_1)$;
$P\, employee(x, y, \text{R\&D}) \vee employee(x, y, \text{R\&D})$.

The first formula is atomic with no free variables, x is free in the second and third ones and x, y are free in the last one. In the third formula, the predicate symbol *employee* occurs both positively and negatively. □

The Semantics

A *first-order finite structure* over the signature $(\mathcal{S}_C, \mathcal{S}_P)$ is a pair (D, I), where D is a countably infinite set, called the *domain*, and I is a *finite interpretation* consisting of an interpretation of constants, $I(c) \in D$, and a finite interpretation of predicate symbols, $I(p) \subseteq D^r$, where r is the arity of p.

A *temporal database* $\mathcal{D}$ is a model structure obtained through the temporalisation of first-order finite structures with respect to the flow $(\mathbb{Z}, <)$, *i.e.* the triple $\mathcal{D} = (\mathbb{Z}, <, g)$, where g associates every time point $t \in \mathbb{Z}$ to a first-order structure $g(t) = (D_t, I_t)$; we further require that $\mathcal{D}$ respects the additional conditions of *constant domains* and *rigid constants*, *i.e.* for every $t, s \in \mathbb{Z}$ and $c \in \mathcal{S}_C$ it must always be the case that:

$D_t = D_s = D$ and
$I_t(c) = I_s(c)$

where the set D is called the *domain of the database*; only the interpretation of predicate symbols is *flexible*, *i.e.* may vary from one time point to another. Note that the symbol "=" is not a database predicate, for it is an infinite relation and it has a rigid interpretation over time. It is usual and very convenient for databases to consider $D = \mathcal{S}_C$, such that I is the identity over constants. Unless otherwise stated, we will assume such a simplification, known as the *Herbrand interpretation* [Lloyd 1987].

A *(global) variable assignment* v is a mapping that associates every variable $x \in V$ to a domain value $v(x) \in D$; the assignment is time independent, so the

variables are treated as global with respect to time. An assignment v' is an *x-variant* of a assignment v if they agree on the values of all variables in V except, possibly, on the value of x.

So let $\mathcal{D} = (\mathbb{Z}, <, g)$ be a temporal database and let v be an assignment. To define the semantics of the formulae of FOTL, it is convenient to extend the assignment over all terms by making $v(c) = I_t(c)$ for $c \in \mathcal{S}_C = D$. We define a formula A to be true in $\mathcal{D}$ at time t under assignment v, writing $\mathcal{D}, v, t \models A$, by induction on the structure of formulae.

$\mathcal{D}, v, t \models p(a_1, \ldots, a_r)$ iff $\langle v(a_1), \ldots, v(a_r)\rangle \in I_t(p)$.

$\mathcal{D}, v, t \models \neg A$ iff it is not the case that $\mathcal{D}, v, t \models A$.

$\mathcal{D}, v, t \models A \wedge B$ iff $\mathcal{D}, v, t \models A$ and $\mathcal{D}, v, t \models B$.

$\mathcal{D}, v, t \models A \vee B$ iff $\mathcal{D}, v, t \models A$ or $\mathcal{D}, v, t \models B$.

$\mathcal{D}, v, t \models S(A, B)$ iff there exists an $s \in T$ with $s < t$ and $\mathcal{D}, v, s \models A$ and for every $u \in T$, whenever $s < u < t$ then $\mathcal{D}, v, u \models B$.

$\mathcal{D}, v, t \models U(A, B)$ iff there exists an $s \in T$ with $t < s$ and $\mathcal{D}, v, s \models A$ and for every $u \in T$, whenever $t < u < s$ then $\mathcal{D}, v, u \models B$.

$\mathcal{D}, v, t \models \exists x A$ iff there exists a v' x-variant of v such that $\mathcal{D}, v', t \models A$.

A temporal database is *temporally bounded* if there are time points t_{max} and t_{min} such that, for all $s < t_{min}$ and $u > t_{max}$, it is the case that $g(s) = g(t_{min})$ and $g(u) = g(t_{max})$. In other words, all the atomic predicates have their truth value "persisting" to the past before t_{min} and to the future after t_{max}. We consider databases to be temporally bounded for the rest of this presentation; this is the first step in trying to guarantee that queries posed to the database will always have finite answers.

Example 4.2 Consider a database $\mathcal{D}$ over the signature of Example 4.1. The countably infinite domain $D = \mathcal{S}_C$ of the database is the set of finite strings built of letters, digits and the symbols '&' and '_'. We take the integer-like flow of time to be that of months, $\{\ldots, Jan92, Feb92, Mar92, \ldots\}$. The (rigid) interpretation of constants is given by $I(c) = c$. Let $I_t(employee)$ be given by

- $\langle$Peter 1K R&D$\rangle \in I_t(employee)$ iff $Jan90 \leq t \leq Dec92$
- $\langle$Peter 2K Marketing$\rangle \in I_t(employee)$ iff $Jan93 \leq t \leq Apr93$

- $\langle$Paul 1K R&D$\rangle \in I_t(employee)$ iff $Jan90 \leq t \leq Dec92$
- $\langle$Mary 3K Finance$\rangle \in I_t(employee)$ iff $Sep91 \leq t \leq Apr93$
- nothing else is in $I_t(employee)$

Note that we have $t_{min} = Dec89$ and $t_{max} = May93$. If we consider a variable assignment v such that $v(x) = Peter$ and $v(y) = 1K$, then

$$\mathcal{D}, v, Apr93 \not\models P\exists y \exists z\, employee(x, y, z) \wedge \neg\exists y_1 \exists z_1\, employee(x, y_1, z_1)$$

for Peter is currently an employee, but

$$\mathcal{D}, v, Apr93 \models P\, employee(x, y, R\&D) \vee employee(x, y, R\&D).$$

for Peter was an employee of the R&D department with salary 1K between $Jan90$ and $Dec92$. □

Data Representation

In non-temporal databases, the issue of data representation never arises because it is an obvious one. But as pointed out by [Kabanza, Stevenne and Wolper 1990], there are uncountably many possible temporal databases and we are therefore limited to finitely representing just a few among those.

Note that we have already limited ourselves the countable class of temporally bounded databases. Temporal data (*i.e.* the coding of FOTL models) will be represented by *temporally labelled formulae* of the form $l : q$, where l is a temporal label and q is an atomic formula. There are several equivalent possibilities for the choice of temporal labels, and two distinct representations will be described here, namely temporal intervals and restricted monadic formulae. For the first case, let $t_0, t_1 \in \mathbb{Z}$ be integer constants; the temporal label l can be the union of intervals of either form

$[t_0, t_1]$ with $t_0 \leq t_1$
$[t_0, +\infty)$
$(-\infty, t_0]$

where, for any $t \in \mathbb{Z}$, $-\infty < t < +\infty$. Examples of temporally labelled formulae are $(-\infty, -3] \cup [7, 23] \cup [72, +\infty) : p_1(a_1)$ and $[5, 5] : p_2(b_1, b_2)$; note that single point intervals, $[t_0, t_0]$, are legal labels. Intervals are always closed unless one of the end points is $-\infty$ or $+\infty$.

Although temporally labelled formulae use the notation of Labelled Deductive Systems (LDS) of [Gabbay 1991a], they are actually representing partial models[1]; a temporally labelled formula $l : p(a_1, \ldots, a_n)$ represents that

$$\langle a_1, \ldots, a_n \rangle \in I_t(p) \text{ for } t \in l.$$

An alternative choice of labels would be to use monadic formulae built from atoms of the form

$$t = t_0 \text{ or } t < t_0 \text{ or } t_0 < t$$

using only the connectives $\wedge$ and $\vee$, such that $t_0 \in \mathbb{Z}$ is any constant and t is the unique variable in the formula and ranges over $\mathbb{Z}$; note that there are no quantifiers in the label. The label $(t_0 \leq t \leq t_1)$ can be used as the obvious abbreviation of $t = t_0 \vee (t_0 < t \wedge t < t_1) \vee t = t_1$. The temporally labelled formula $l(t) : p(a_1, \ldots, a_n)$ represents that

$$\langle a_1, \ldots, a_n \rangle \in I_t(p) \text{ for } t \text{ satisfying } l(t) \text{ over } \mathbb{Z}.$$

$t_0 : q$ is used as an obvious abbreviation for $[t_0, t_0] : q$ and $t = t_0 : q$.

Given a finite database representation, *i.e.* a finite set of temporally labelled formula, it is assumed that the union of the represented partial models constitutes a (total) model; this is a model theoretic counterpart of the syntactic notion of the Closed World Assumption [Reiter 1984].

Example 4.3 The database of example 4.2 is represented by

- $(Jan90 \leq t \leq Dec92) : employee(\text{Peter}, \text{1K}, \text{R\&D})$
- $(Jan93 \leq t \leq Apr93) : employee(\text{Peter}, \text{2K}, \text{Marketing})$
- $(Jan90 \leq t \leq Dec92) : employee(\text{Paul}, \text{1K}, \text{R\&D})$
- $(Sep91 \leq t \leq Apr93) : employee(\text{Mary}, \text{3K}, \text{Finance})$

□

[1] for partial models, it is perhaps more intuitive to think in terms of the equivalent definition of predicate interpretation, $I : \mathcal{S}_P \rightarrow D^n \times 2^{\mathbb{Z}}$

Observation

In the temporal database literature, it is common to find the temporal relations restricted to a *temporal normal form (TNF)* [Navathe and Ahmed 1988]. This normal form assumes that there are no two tuples in a relation with identical key attribute(s) associated to overlapping intervals. For example, suppose that $t_1 \leq t_2 \leq t_3 \leq t_4$; then the following representation would be violating the TNF, even if $t_2 = t_3$:

$$(t_1 \leq t \leq t_3) : q$$
$$(t_2 \leq t \leq t_4) : q$$

In the presentation of the subsequent examples, as in the previous ones, this normal form will be obeyed, although none of our results actually depends on the existence of the TNF. It should not be difficult to see that non-TNF representations can be brought to an equivalent TNF one, *e.g.* in the example above $(t_1 \leq t \leq t_4) : q$ is a TNF representation of the same temporal relation.

Queries

The fact that we want the result of queries to be *finite* relations forces us to restrict the format of the formulae that are acceptable as queries. For that, let us first define the notion of a *relevant* domain element. An element d in the domain D is relevant to a predicate symbol $p \in \mathcal{S}_P$ if it occurs in $I_t(p)$, for some $t \in \mathbb{Z}$; d is relevant to a formula A if it is a constant occurring in A or it is relevant to some predicate symbol occurring in A; let $R_{\mathcal{D}} \subset D$ be the set of all elements relevant to the predicate symbols in the database signature, which clearly is a finite set.

A *domain independent* formula $A(x_1, \ldots, x_n)$ is one whose interpretation generates a finite n-place relation containing only domain elements that are relevant to it, *i.e.* for every time t the set of tuples of domain elements $\langle v(x_1), \ldots, v(x_n) \rangle$ such that $\mathcal{D}, v, t \models A(x_1, \ldots, x_n)$, is finite and contains only elements that are relevant to $A(x_1, \ldots, x_n)$. Ideally, an acceptable query should be a *domain independent* formula, but unfortunately it is an undecidable problem to tell whether a formula is domain independent [Ullman 1988]. Thus we syntactically define the class of *safe* formulae below as an alternative sufficient condition to obtain domain independence. The basic idea behind safeness is that of "limiting" all free variables that appear in disjunctions, negations and temporal subformulae.

For disjunctions $A \vee B$, it is simply required that A and B share the same free variables, for if we have the non-safe formula $A(x) \vee B(y)$ such that $A(x)$ holds

for some domain element $x_0 \in D$, there are infinitely many pairs (x_0, d), $d \in D$, satisfying the query.

For negations, the idea of limiting a variable is similar to that of *range restricted* clauses in logic programming [Lloyd 1987]. Basically, all free variables inside a negation are required to occur positively outside the negation. For example, the formula $\neg A(x, y)$ is not safe, but in $\neg A(x, y) \wedge B(x) \wedge C(y)$ the free variable of negated A are limited due to their positive occurrence in B and C.

An extra safeness problem occurs with temporal formulae: the temporal formula $S(A(x), B(y))$ is not safe, because if $A(x)$ is true at the previous moment for some $x_0 \in D$, there are infinitely many pairs (x_0, d), $d \in D$, satisfying the formula; the formula $S(A(x) \wedge B(y), B(y))$[2], however, does not present that problem and it is considered safe; that is how the semantics of the S operator is defined in the US-based temporal algebra in [Gabbay and McBrien 1991] in terms of the semantics we present here. In general, for temporal formulae of the form $S(A, B)$ and $U(A, B)$ to be safe it is required that:

(a) free variables that are limited inside A, are also considered limited in $S(A, B)$ and $U(A, B)$;

(b) all free variables occurring in B have to be limited outside B.

The variable y is not limited outside $B(y)$ in the non-safe formula $S(A(x), B(y))$, but it is limited in safe formulae as $S(A(x) \wedge B(y), B(y))$ and $S(A(x), B(y)) \wedge C(y)$.

We follow Ullman's [1988] formal presentation of safe formulae for non-temporal databases; further discussions on safeness can be found in [Zaniolo 1986; Ramakrishnan, Bancilhon and Silberschatz 1987]. But, before we present the formal definition, just a small remark: a subformula X is a maximal conjunction in a formula A if X is a subformula that is not part of a conjunction. For example, in the formula $(\neg(A \wedge B \wedge C) \vee S(A, B)) \wedge E$ there are six maximal conjunction subformulae, namely $A \wedge B \wedge C$, $\neg(A \wedge B \wedge C)$, $S(A, B)$, A, B and the whole formula.

Definition 4.1 Safe queries A formula is safe when:

(a) If it contains a subformula that is the disjunction of B_1 and B_2, then B_1 and B_2 share the same free variables, *i.e.* the subformula is of the form $B_1(x_1, \ldots, x_n) \vee B_2(x_1, \ldots, x_n)$.

[2] this is equivalent to define the semantics of $S(A, B)$ as

$\mathcal{D}, v, t \models S(A, B)$ iff $\exists s < t$ and $\mathcal{D}, v, s \models A$ and $\forall u$, $s \leq u < t$ implies $\mathcal{D}, v, u \models B$.

and similarly for $U(A, B)$.

(b) If it contains a subformula that is a maximal conjunction $B_1 \wedge \ldots \wedge B_m$, then all the free variables appearing in the B_i's must be limited in the following sense.

- a variable is limited if it occurs in some B_i, where B_i is not an equality nor is it negated nor temporal;
- if B_i is of the form $x = c$ or $c = x$, where c is a constant, then x is limited;
- if B_i is of the form $x = y$ or $y = x$, where y is a limited variable, then x is limited;
- if B_i is of the form $S(A, C)$ or $U(A, C)$, then, recursively, all free variables that are limited in A are limited in B_i.

(c) It contains a subformula of the form $\neg B(x_1, \ldots, x_k)$ only in the terms of (b), *i.e.* $\neg B(x_1, \ldots, x_k)$ must be part of a conjunction or temporal formula such that all x_i are limited.

(d) It contains a subformula of the form $S(A, B)$ or $U(A, B)$ only in the terms of (b), *i.e.* $S(A, B)$ or $U(A, B)$ must be part of a maximal conjunction such that all the free variables are limited.

A *(safe) query* Q is then represented by $Q = \{x_1, \ldots, x_m; t \mid A(x_1, \ldots, x_m)\}$, where $A(x_1, \ldots, x_m)$ is a safe formula with free variables $x_1, \ldots, x_m$; the *snapshot relation generated by the query Q on the database $\mathcal{D}$* is $\{\langle v(x_1), \ldots, v(x_m)\rangle \mid$ there exists v such that $\mathcal{D}, v, t \models A(x_1, \ldots, x_m)\}$; each tuple in the generated relation is said to *satisfy* the query Q at time t. A temporal query is *domain independent* if, for every $t \in \mathbb{Z}$, it generates only finite snapshot relations. □

If we want queries to retrieve times as well, we can define the *temporal relation* generated by a query $Q = \{x_1, \ldots, x_m \mid A(x_1, \ldots, x_m)\}$ as the set of labelled tuples $t_0 : \langle v(x_1), \ldots, v(x_m)\rangle$, $t_0 \in \mathbb{Z}$, such that there exists v, $\mathcal{D}, v, t_0 \models A(x_1, \ldots, x_m)$[3]. If the database is time bounded and a query is domain independent it follows that the full temporal relation it generates can be finitely represented with the labelled tuples of the format previously described.

Proposition 4.1 *Safe queries are domain independent.*

[3]Note that in our query language there is no reference to time points, so in order to take advantadge of full temporal relations it would be necessary to introduce time referrences in the temporal operators language, as it was done in the TEMPORA ERL-language [McBrien *et al.* 1991].

Proof We know from [Ullman 1988] that every non-temporal subformulae with limited variables generate only finite relations over relevant domain elements, and so do safe disjunctions, thus this is the case for any time t; in fact, this may be derived from Codd's original result on the equivalence of the (finitely based) relational algebra and the relational calculus [Codd 1970; 1972]. With regards to safe temporal subformulae of the form $S(A, B)$ and $U(A, B)$, A can only generate finite relations over relevant domain elements at any time t, and B has all its free variables limited; since the database is temporally bounded, only finitely many tuples of domain elements (corresponding to the free variables) can satisfy $S(A, B)$ and $U(A, B)$ at every time point. So temporal subformulae can only generate finite snapshot relations over relevant domain elements and safe temporal formulae are domain independent. □

Note that we may have non-safe queries that are domain independent. For instance, the formula

$$A(x, y, z) \wedge \neg(FB(x, y) \vee C(y, z))$$

is non-safe, but it generates only finite relations over $R_{\mathcal{D}}$ on databases because it is logically equivalent to the safe formula

$$A(x, y, z) \wedge \neg FB(x, y) \wedge \neg C(y, z).$$

In fact, the temporal formula $GA(x)$, defined as $\neg S(\neg A(x), \top)$, is not safe, but it is equivalent over $\mathbb{Z}$ to the safe formula $\bigcirc(A(x) \wedge GA(x))$.

Example 4.4 In the database of Example 4.2, if we want to know the names of employees that were sacked in the past, as of *Apr*93, we can pose the following safe query

$$\{x; Apr93 \mid P\exists y \exists z\, employee(x, y, z) \wedge \neg\exists w \exists v\, employee(x, w, v)\}$$

which generates the one-place unary relation $\{\langle \text{Paul} \rangle\}$. If we want to know the name and salaries of employees that have ever worked in the R&D department, as of *Apr*93, we pose the following safe query to the database

$$\{x, y; Apr93 \mid P\, employee(x, y, \text{R\&D}) \vee employee(x, y, \text{R\&D})\}$$

which generates the relation $\{\langle \text{Peter 1K} \rangle, \langle \text{Paul 1K} \rangle\}$. □

A brief comment on complexity issues is made here. Even though a logic is undecidable, it can still be used to efficiently compute safe queries. For example, first-order logic is undecidable but safe first-order queries are computed in polynomial time. Full first-order temporal logic cannot even be finitely axiomatised over $\mathbb{Z}$, but safe temporal queries can be computed in polynomial time too; see [Gabbay and McBrien 1991; Tuzhilin and Clifford 1990; McBrien 1992]. In this work, the two-dimensional temporal model will not be considered for the purposes of query evaluation; the results of the previous chapters will show their usefulness in the discussion on database updates.

4.2 Propositional Abstractions

The first-order approach of the previous section differs from the propositional treatment of temporal features in the previous chapters. To reconcile these two different approaches this section presents a propositional abstraction of database queries.

Let D be the database domain and let $R_{\mathcal{D}} \subset D$ be the finite set of domain elements that are relevant to the predicate symbols in the database signature $\mathcal{S}_P$. We define the propositional signature $\mathcal{P}$ abstracting from $\mathcal{S}$ as consisting of the following propositional atoms:

- $[\![a = b]\!]$, for each $a, b \in D$;
- $[\![p(a_1, \ldots, a_{ar(p)})]\!]$, for each $p \in \mathcal{S}_P$ and each $a_i \in D$.

It is no coincidence that we choose first-order $[\![]\!]$-enclosed atoms to represent propositions; the first item above allows us to equate two constant symbols, a constant symbol with a domain element, and two domain elements (this latter equality when relating to two distinct domain elements will actually generate propositions that are always false, in the same way that equating two identical symbols will generate propositions that are always true); the second item above generates a proposition for each possible ground predicate. The generalisation of this notation will give us a propositional abstraction of FOTL-formulae. If v is an assignment, let B^v be a *v-grounded formula* obtained by substituting every free variable x occurring in B by $v(x)$. For every safe FOTL-formula A, we define its *propositional abstraction* with respect to v by taking $B = A^v$ and generalising the above propositional notation over v-grounded formulae, denoted by $[\![B]\!]$, where:

- $[\![\neg B]\!] = \neg [\![B]\!]$;

- $[\![B_1 \wedge B_2]\!] = [\![B_1]\!] \wedge [\![B_2]\!]$;
- $[\![S(B_1, B_2)]\!] = S([\![B_1]\!], [\![B_2]\!])$;
- $[\![U(B_1, B_2)]\!] = U([\![B_1]\!], [\![B_2]\!])$;
- $[\![\exists x B]\!] = \bigvee_{d \in R_{\mathcal{D}}} [\![B(x \setminus d)]\!]$.

where $B(x \setminus d)$ is the formula obtained by substituting all free occurrences of x in B by d; the last item above is a well defined propositional formula because $R_{\mathcal{D}}$ is a finite set.

Definition 4.2 Given a database $\mathcal{D} = (\mathbb{Z}, <, g)$ over a signature $\mathcal{S} = (\mathcal{S}_C, \mathcal{S}_P)$, we say that a propositional model $\mathcal{M} = (\mathbb{Z}, <, h)$ over $\mathcal{P}$ *abstracts from* $\mathcal{D}$ if for every assignment v extended over constant symbols, and every $t \in \mathbb{Z}$, it is the case that for every atomic formula of the form $p(a_1, \ldots, a_r)$,

$$[\![p(a_1, \ldots, a_r)]\!] \in h(t) \quad \text{iff} \quad \langle v(a_1), \ldots, v(a_r) \rangle \in I_t(p);$$

and for every atomic formula of the form $a = b$

$$[\![a = b]\!] \in h(t) \quad \text{iff} \quad v(a) = v(b).$$

□

A straightforward induction on the structure of formulae then shows that:

$$\mathcal{D}, v, t \models B(x_1, \ldots, x_m) \quad \text{iff} \quad \mathcal{M}, t \models [\![B^v]\!].$$

The relation generated by the safe formula $B(x_1, \ldots, x_m)$ can be expressed as

$$\{\langle v(x_1), \ldots, v(x_m) \rangle \mid \text{ there exists } v \text{ such that } \mathcal{M}, t \models [\![B(x_1 \setminus v(x_1), \ldots, x_m \setminus v(x_m))]\!]\}.$$

The propositional abstraction as defined above has nothing especially "temporal" about it, the whole purpose of it being the elimination of variables and quantifiers from safe formulae. However, it is important for sending us back to the propositional framework. From now on, we refer to the contents of a database by its propositional abstraction $\mathcal{M}$; moreover, we can refer to the update of ground atomic information in a database as the update of propositional atoms. We assume that the countably infinite domain D of the database remains the same after the update, and so does the database signature.

Example 4.5 Consider the database $\mathcal{D}$ from Example 4.2 (where $R_{\mathcal{D}} \subset D = \mathcal{S}_C$); let v an assignment such that $v(x) = Peter$ and $v(y) = 10\text{K} \notin R_{\mathcal{D}}$. A few of the properties of the database propositional abstraction model $\mathcal{M} = (\mathbb{Z}, <, h)$ are

- $[\![employee(\text{Peter}, 1\text{K}, \text{R\&D})]\!] \in h(t) \quad \text{iff} \quad Jan90 \leq t \leq Dec92$;
- $[\![employee(\text{Peter}, \text{y}, \text{R\&D})^v]\!] \notin h(t)$, for every $t \in \mathbb{Z}$.

Consider the following safe formula about sacked employees

$$A(x) = P\exists y \exists z\, employee(x, y, z) \wedge \neg \exists y \exists z\, employee(x, y, z).$$

and its propositional abstraction under v:

$$[\![A(x)^v]\!] = P\left(\bigvee_{c \in R_{\mathcal{D}}} \bigvee_{d \in R_{\mathcal{D}}} [\![employee(Peter, c, d)]\!]\right) \wedge \neg\left(\bigvee_{c \in R_{\mathcal{D}}} \bigvee_{d \in R_{\mathcal{D}}} [\![employee(Peter, c, d)]\!]\right).$$

It follows that, for all $t \in \mathbb{Z}$, $\mathcal{D}, v, t \models A(x) \quad \text{iff} \quad \mathcal{M}, t \models P[\![A(x)^v]\!]$. □

Notation: We may sometimes abuse the notation and represent the propositional abstraction of a predicate formula by the formula itself. We do this when no ambiguity is implied, mainly when we refer to atomic formulae with no free variables.

4.3 A Two-dimensional Description of Database Evolution

In describing the evolution of a temporal database, we have to distinguish the database evolution from the evolution of the world it describes. The "world", also called the *Universe of Discourse*, is understood to be any particular set of objects in a certain environment that we may wish to describe. The database, in its turn, contains a description of the world. Conceptually, we have to bear in mind two distinct types of evolution, as we introduced in [Finger 1992]:

- The *evolution of the modelled world* is the result of changes in the world that occur independently of the database.
- A temporal database contains a description of the history of the modelled world that is also constantly changing due to database updates, generating a sequence of database states. This *evolution of the temporal description* does

not depend only on what is happening at the present; changes in the way the past is viewed also alter this historical description; moreover, changes in expectations about the future, if those expectations are recorded in the database, also generate an alteration of the historical description. This process is also called *historical revision*.

These two distinct concepts of evolution are reflected by a distinction between two kinds of flows of time, whether their time points refer to a moment in the history of the world, or whether they are associated to a moment in time at which a historical description is in the database.

Several different names are found in the literature for these two time concepts. The former is called *evaluation time* [Kamp 1971; Gabbay, Hodkinson and Reynolds 1994], *historical time* [Finger 1992], *valid time* [Snodgrass and Ahn 1985] and *event time* [McKenzie and Snodgrass 1991]. The latter time concept is called *utterance time* [Kamp 1971], *reference time* [Gabbay, Hodkinson and Reynolds 1994], *transaction time* [Finger 1992; Snodgrass and Ahn 1985] and *belief time* [Sripada 1990]. In this presentation we chose to follow a glossary of temporal database concepts proposed in [Jensen *et al.* 1992], calling the former valid-time, which is associated to the horizontal dimension in our two-dimensional model, and calling the latter transaction-time, which is associated to the vertical dimension.

So we use the two-dimensional plane model to simultaneously cope with the two notions of time in the description of the evolution of a temporal database, as illustrated in Figure 4.1.

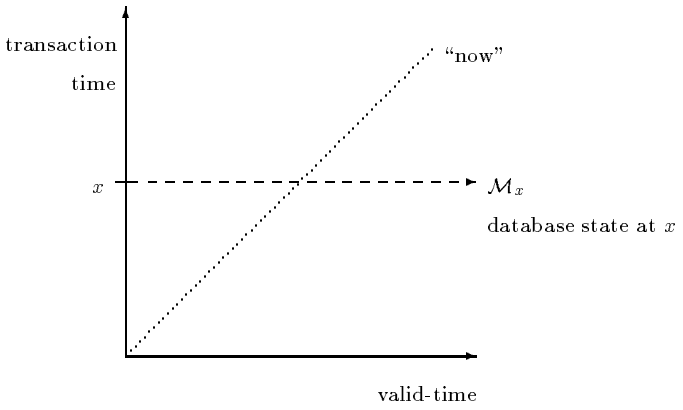


Figure 4.1 Two-dimensional database evolution

Let $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$ be a two-dimensional plane model over $\mathcal{K} \times \overline{\mathcal{K}}$; its *horizontal projection* with respect to the vertical point $x \in \overline{T}$ is the one-dimensional temporal model

$$\mathcal{M}_x = (T, <, g_x),$$

such that, for every propositional atom q, time points $t \in T$ and $x \in \overline{T}$,

$$q \in g_x(t) \quad \text{iff} \quad q \in g(t, x).$$

It follows that for every horizontal US-formula A and for every $t \in T$ and $x \in \overline{T}$,

$$\mathcal{M}_x, t \models A \quad \text{iff} \quad \mathcal{M}, t, x \models A.$$

The horizontal projection represents a state of a temporal database. According to our convention, we have that $\mathcal{K} = \{\mathbb{Z}\}$; and since we are interested in describing the evolution of database states as a linear and discrete process, we also fix $\overline{\mathcal{K}} = \{\mathbb{Z}\}$.

Updating temporal databases requires that, besides specifying the atom to be inserted or deleted, we specify the time where the atom is to be inserted or deleted. For that reason, it is convenient to use the notation of temporally labelled formulae to represent the data being inserted and deleted; in most cases, we will restrict our attention to labelled atoms only. Labelled formulae allow for finite representation of possibly infinite information, but for update purposes we will consider possibly infinite sets of *restricted labelled formulae* of the form $t_0 : q$, $t_0 \in \mathbb{Z}$; eventually, these sets will be replaced by those representable by finite sets of temporally labelled formulae as defined in Section 4.1.

An *update pair* (θ_+, θ_-) consists of two disjoint sets of restricted labelled atoms, where θ_+ is the *insertion set* and θ_- is the *deletion set*. We say that an update pair determines or characterises a *database update* Θ_x occurring at transaction time $x \in \mathbb{Z}$ if the application of the update function Θ_x to the database state $\mathcal{M}_x = (T, <, g_x)$ generates a database state $\Theta_x(\mathcal{M}_x) = (T, <, \Theta_x(g_x))$ satisfying, for every propositional atom q and every time point $t_0 \in T$,

- if $t_0 : q \in \theta_+$, then $q \in \Theta_x(g_x)(t_0)$;
- if $t_0 : q \in \theta_-$, then $q \notin \Theta_x(g_x)(t_0)$;
- if neither $t_0 : q \in \theta_+$ nor $t_0 : q \in \theta_-$, then $q \in \Theta_x(g_x)(t_0) \quad \text{iff} \quad q \in g_x(t_0)$.

The first item corresponds to the insertion of atomic information, the second one corresponds to the deletion of atomic information, and the third one corresponds to the persistency of the unaffected atoms in the database. Note that only a finite number of propositional atoms can occur in θ_+ and θ_- because there are only finitely many atoms in the database. The update Θ_x is a database state transformation

function. An update may be empty ($\theta_+ = \theta_- = \varnothing$), in which case the transformation function is just the identity and the database state remains the same.

Let Θ_x be a database update characterised by the pair (θ_+, θ_-). It is a *bounded* update if there exist $t', t'' \in \mathbb{Z}$ such that for every atom q in θ_+ (resp. q in θ_-),

- for all $t_0 > t'$, $t_0 : q \in \theta_+$ (resp. $t_0 : q \in \theta_-$) iff $t' : q \in \theta_+$ (resp. $t' : q \in \theta_-$); and
- for all $t_0 < t''$ $t_0 : q \in \theta_+$ (resp. $t_0 : q \in \theta_-$) iff $t'' : q \in \theta_+$ (resp. $t'' : q \in \theta_-$).

A sequence of database updates $\{\Theta_x\}_{x \in \mathbb{Z}}$ is said to be *bounded* if every database update Θ_x is bounded.

We say that a two-dimensional plane model $\mathcal{M}$ *represents the evolution in time of a temporal database* through the update sequence $\{\Theta_x\}_{x \in \mathbb{Z}}$ if, for every $x \in \mathbb{Z}$, $\Theta_x(\mathcal{M}_x) = \mathcal{M}_{x+1}$.

Proposition 4.2 *Let $\mathcal{M}$ be a two-dimensional model representing the evolution a temporal database through the update sequence $\{\Theta_x\}_{x \in \mathbb{Z}}$ with initial time x_0, such that $\mathcal{M}_{x_0}$ is temporally bounded. Then, for every $x \geq x_0$, $\mathcal{M}_x$ is temporally bounded iff $\{\Theta_x\}_{x \in \mathbb{Z}}$ is bounded.*

Proof The two directions of the iff-condition are proved separately.

($\Leftarrow$) If Θ_x is a bounded database update, there exists a time t' after which, for every atom q, either its truth value is determined and equal for all times, or its truth value is not affected by the update; similarly towards the past. Therefore, if Θ_x is applied to a temporally bound database state it generates another temporally bounded database state.

($\Rightarrow$) Suppose $\mathcal{M}_x$ and $\Theta_x(\mathcal{M}_x) = \mathcal{M}_{x+1}$ are both temporally bounded. Then either there is only finite amount of labelled formulae $t : q$ changing its value from one state to the next, in which case Θ_x is bounded, or there are atoms q_i such that their value has changed in finitely many times. In this last case, since both $\mathcal{M}_x$ and $\mathcal{M}_{x+1}$ are temporally bounded, there must be times t' and t'' after which and before which, respectively, $t : q$ was always inserted or always deleted. Therefore Θ_x is bounded. □

The Proposition above shows us that bounded updates are the kind of update we want to allow in temporally bounded databases, for which the two-dimensional models describe the evolution. Bounded updates are easily shown to be finitely representable by the labelled formulae of Section 4.1, as shown by the following example.

Example 4.6 Consider the evolution monthly evolution of the database of Example 4.2. Suppose that at $Apr93$ we decide to retroactively increase Mary's monthly salary to 5K for the whole year, which is illustrated in Figure 4.2.

This situation is described by an update $\Theta_{Apr93} = (\theta_+, \theta_-)$, where

$$\theta_+ = \{[Jan93, Dec93] : employee(\text{Mary}, 5\text{K}, \text{Finance})\}$$
$$\theta_- = \{[Jan93, Dec93] : employee(\text{Mary}, 3\text{K}, \text{Finance})\}$$

Note that we specified the deletion of Mary's old salary until $Dec93$; the same effect would be achieved had we only specified the deletions until $Apr93$, *i.e.* the same horizontal projection would have been generated for $\mathcal{M}_{May93}$. Note also that we have changed not only the past, but also Mary's salary at the present, $Apr93$, and also its expectation for the future.

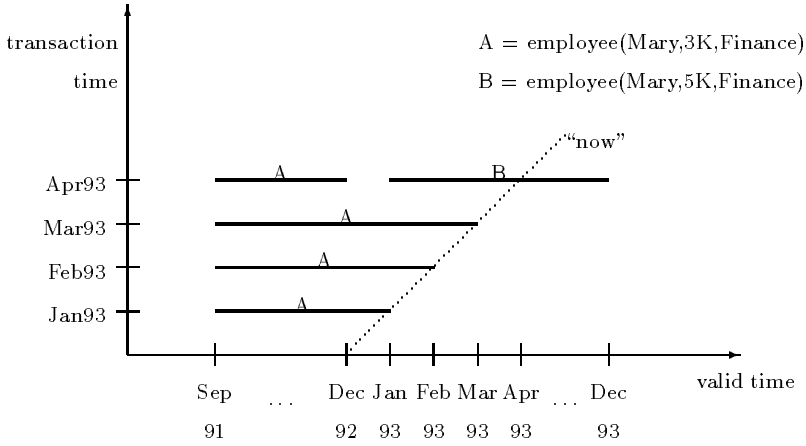


Figure 4.2 Two-dimensional diagram of database evolution

If we wanted to increase Mary's salary indefinitely to the future, we could have used the following infinite update $\Theta_{Apr93} = (\theta_+, \theta_-)$, where

$$\theta_+ = \{[Jan93, +\infty) : employee(\text{Mary}, 5\text{K}, \text{Finance})\}$$
$$\theta_- = \{[Jan93, Dec93] : employee(\text{Mary}, 3\text{K}, \text{Finance})\}$$

This illustrates the result of Proposition 4.2, where time boundedness is preserved by this infinite update. □

4.4 Valid-time and Transaction-time Databases

In the previous section we pointed out a conceptual difference between the valid-time and transaction-time flows of time, and treated them separately but simultaneously

in the two-dimensional model of the evolution of a temporal database. For that, we assumed that the database records the valid-time flow of time; however, it need not be so.

In their taxonomy of time in databases, Snodgrass and Ahn [1985] distinguish a *historical database* (called here a *valid-time database*), a temporal database that records only valid time, from a *rollback database* (called here a *transaction-time database*), a database that records only transaction time. The temporal data in a valid-time database is supposed to describe the evolution of the modelled part of the world that occurs independently from the database; on the other hand, the temporal information associated with any piece of data recorded in a transaction-time database is generated automatically and is supposed to represent the times when that information held in the database; therefore, transaction time has no existence independent from the database. For example, suppose that at (transaction time) $Sep91$ the piece of data $employee(\text{Mary}, 3\text{K}, \text{Finance})$ is inserted in the transaction-time database; this information will persist for all times t, from t starting at $Sep91$, until the time that data is deleted from the database. Suppose the deletion happens at (transaction time) $Apr93$, so that in the database model we have

$$[Sep91, Apr93] : employee(\text{Mary},\ 3\text{K},\ \text{Finance}).$$

Note that the temporal information was generated automatically at the times of insertion and deletion, without the need to externally supply them. If we want then, at $Apr93$, to increase Mary's salary retroactively to the beginning of the year, we cannot record this fact in the transaction-time database. The transaction-time database records only a sequence of (non-temporal) database states, attributing to each state a (transaction) time-stamp, thus allowing us to reconstruct a previous state but not to modify it.

No present and past query that may be posed to a temporal database can tell whether it is a valid-time or a transaction-time database. In fact, the query language and the notion of a correct answer (the generated relation) for a query are exactly the same in both cases. Even if we do not have in the database any information concerning a time later than the current time, based on the fact that transaction-time databases record only the present and past states of the database, we cannot guarantee that the given temporal database is a transaction-time one, for it is perfectly legal for a valid-time database to store only data about the present and past.

The difference between valid-time and transaction-time databases lies in their dynamic behaviour, not in their static query answering. We still consider the state of any temporal database to be the horizontal projection of a two-dimensional plane

model with respect to some vertical (hence transaction-time) point. An *tt-update* (transaction time update) is a state transformation function Θ_x, where x is the current (transaction-) time, determined by the update pair (θ_+, θ_-) satisfying the following conditions.

(a) the formulae of θ_+ and θ_- are of the form $y : q$ where $y \geq x$, *i.e.* there are no updates in the past;

(b) $x : q \in \theta_+$ (resp. $x : q \in \theta_-$) iff for all $y \geq x$, $y : q \in \theta_+$ (resp. $y : q \in \theta_-$); *i.e.* updates persist to the future.

We can then use the two-dimensional plane model to describe the evolution of a transaction-time database so that we may characterise it by the properties of the two-dimensional plane model that describe its evolution. The basic distinction between valid-time and transaction-time databases is that in transaction-time databases we cannot change the past, whereas in the valid-time database any change is allowed. To characterise this impossibility to change the past, we will have to first characterise the existence of a "now" in the two-dimensional plane model.

Recall that in Section 3.5 we used a special propositional symbol δ to characterise the "diagonal" of a two-dimensional model $\mathcal{M}$; the formulae

D1 $\Diamond\delta \wedge \overline{\Diamond}\delta$

D2 $\delta \rightarrow (G\neg\delta \wedge H\neg\delta \wedge \overline{G}\neg\delta \wedge \overline{H}\neg\delta)$

D3 $\delta \rightarrow (\overline{H}G\neg\delta \wedge \overline{G}H\neg\delta)$

are valid over a two-dimensional plane model over $\mathcal{K}\times\mathcal{K}$ iff there is an isomorphism between the horizontal and vertical flows of time. In the case of a model $\mathcal{M}$ over $\mathbb{Z}\times\mathbb{Z}$, we can take the identity of points over the horizontal and vertical flows of time as the obvious choice of isomorphism, so that the formula $\overline{\Box}\Box(\mathbf{D1}\wedge\mathbf{D2}\wedge\mathbf{D3})$ is valid iff, for all $t \in \mathbb{Z}$,

$$\mathcal{M}, t, t \models \delta.$$

The points where δ holds are exactly those where the valid- and transaction-times coincide, so we use those diagonal points as the ones where "now" holds. The formula $\mathbf{D1}\wedge\mathbf{D2}\wedge\mathbf{D3}$ belongs to the fully combined language of US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$, but it does not belong to the partially interlaced language of $\mathsf{US}\times\bar{\mathsf{N}}\bar{\mathsf{P}}$. This problem can be solved by adopting the formulae

d1 $\Diamond\delta$

d2 $\delta \rightarrow (G\neg\delta \wedge H\neg\delta)$

d3 $\delta \leftrightarrow \overline{\bigcirc}\bigcirc\delta$

that are in the language of $\mathsf{US}\times\bar{\mathsf{N}}\bar{\mathsf{P}}$ over $\mathbb{Z}\times\mathbb{Z}$. We define a formula A to *hold over* a two-dimensional plane model $\mathcal{M} = (T, <, \overline{T}, \overline{<}, g)$, which is represented by $\mathcal{M} \models A$, if for every $t \in T$ and $x \in \overline{T}$, $\mathcal{M}, t, x \models A$. Then the following result gives us the desired equivalence; the proof is in Appendix B, Lemma B.2.

Lemma 4.1 *Let $\mathcal{M}$ be a two-dimensional plane model over $\mathbb{Z}\times\mathbb{Z}$. Then the formula* **D1∧D2∧D3** *holds over $\mathcal{M}$ iff* **d1∧d2∧d3** *holds over $\mathcal{M}$.*

To characterise the persistence of present data towards the future, we make the one-dimensional formula

Persist $(\delta\wedge q)\rightarrow Gq$,

hold over a two-dimensional model $\mathcal{M}$ for every literal q, where a literal is an atom or the negation of an atom.

The "no change in the past" feature of transaction-time databases is characterised as the persistence of atomic information of the "now", *i.e.* on the diagonal, towards the vertical future. Therefore, the following formula must hold over two-dimensional plane models that represent the evolution of a temporal database

Roll $((\delta\vee F\delta)\wedge q)\rightarrow\overline{\bigcirc}q$,

where q is any literal. Note that the formula above belongs to all languages previously mentioned. The subformula $(\delta\vee F\delta)$ represents the fact that we are in the present or past, so whatever information we have then will persist to the next vertical moment, when it will certainly be part of the (horizontal, therefore the database's) past. By making such a formula hold over the two-dimensional plane model, we guarantee the persistence of the information about the past in all states of the database. The following property generalises **Roll** for a larger class of formulae; the proof is in Appendix B, Lemma B.3.

Lemma 4.2 *If* **Roll** *holds over a two-dimensional $\mathcal{M}$ for any literal q, it also holds over $\mathcal{M}$ for any* US*-formula that does not contain future operators, i.e. does not contain U and its derived operators.*

For the languages resulting from the full combination of US and $\bar{\mathsf{U}}\bar{\mathsf{S}}$,another version of **Roll** is possible, namely if the following holds over a two-dimensional plane model $\mathcal{M}$ for any literal q

Roll2 $\overline{P}(\delta\wedge q)\rightarrow q$.

It actually expresses that whatever was true on the diagonal remains true; **Roll2** can also be extended over $\mathcal{M}$ for any past-present US- formula.

The characterisation of transaction-time databases as "no updates in the past" is given by the following.

Proposition 4.3 *Let $\mathcal{M}$ be a two-dimensional model representing the evolution of a temporal database through the update sequence $\{\Theta_x\}_{x\in\mathbb{Z}}$ such that* **Persist** *holds over $\mathcal{M}$. Then the following are equivalent:*

(a) every update Θ_x is a tt-update;

(b) **Roll** *holds over $\mathcal{M}$, for the fully combined language of $\mathsf{US}\times\bar{\mathsf{U}}\bar{\mathsf{S}}$, for the fully combined language of $\mathsf{US}\times\bar{\mathsf{N}}\bar{\mathsf{P}}$.*

(c) **Roll2** *holds over $\mathcal{M}$, for the fully combined language of $\mathsf{US}\times\bar{\mathsf{U}}\bar{\mathsf{S}}$.*

Proof (a $\Rightarrow$ b) A *tt*-update does not update the past, so for $t\leq x$, by the semantics of two-dimensional updates, the atomic information in $\mathcal{M}$ at t,t persist into the vertical, transaction-time future. So **Roll** holds over $\mathcal{M}$.

(b $\Rightarrow$ c) We need to consider only $t\leq x$. Assume **Roll** holds over $\mathcal{M}$; a simple induction on $x-t$ shows that $\mathcal{M},t,x\models q$ iff $\mathcal{M},t,t\models q$, for any literal q. So **Roll2** holds over $\mathcal{M}$.

(c $\Rightarrow$ a). Assume that (c) holds and suppose that at transaction time x there was an update in the past time $t<x$. Without loss of generality, suppose it was an insertion, so that for some atom q, $\mathcal{M},t,x\models\neg q$ and $\mathcal{M},t,x+1\models q$. If we concentrate on the diagonal, if $\mathcal{M},t,t\models q$ then $\overline{P}(q\wedge\delta)\rightarrow q$ fails at (t,x), and if $\mathcal{M},t,t\models\neg q$ then $\overline{P}(q\wedge\delta)\rightarrow q$ fails at $(t,x+1)$, contradicting (c). □

The formulae **Persist**, **Roll** and **Roll2** are not axioms in the sense that were used in the previous chapters. This formulae are actually meta-level constraints on a two-dimensional model $\mathcal{M}$ (in fact, they are even second order, for they require that some property hold "for all literals").

It follows that a transaction-time database is one whose evolution is described by a two-dimensional model $\mathcal{M}$ such that **Persist** and **Roll** hold over $\mathcal{M}$. The persistence of information on a transaction-time database is illustrated in Figure 4.3. It is no coincidence that there is a diagonal symmetry in Figure 4.3, where it can be seen that a literal q that holds at the diagonal (*i.e.* at some current time) persists into the horizontal future in the current database state, and also persists into the vertical future throughout all the future database states, when it will be part of the unmodifiable past.

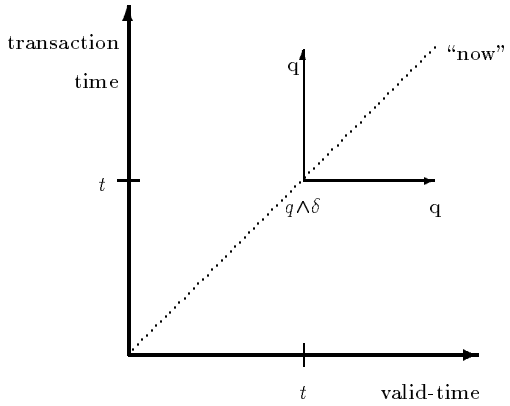


Figure 4.3 Persistence of atomic data in transaction time databases

A temporal database is a valid-time one if it is possible that a two- dimensional model describing its evolution does not satisfy the three "meta-level" axioms above. That does not mean that all two- dimensional models describing the evolution of a valid-time database invalidate all those meta-level axioms, because it is possible for a valid-time database to behave like a transaction- time one, *i.e.* the valid-time database can simulate a transaction-time one.

Chapter 5

Detection of Time Paradoxes in Temporal Active Databases

In this chapter we show how to create logical links between information associated to possibly distinct times in history and we discuss the effects that changes in history may have upon those temporal links.

For the purpose of establishing those links, we extend a valid-time database with *temporal rules*, and we provide those rules with an execution semantics; the resulting combined system is called an *active valid-time database*.

Besides the execution semantics, we define a declarative *valid-time interpretation* of the rules in an active database. We show that, under the execution semantics, the occurrence of updates at any time may cause an invalidation of the valid-time interpretation of rules the at some time in the database state, generating a *time paradox*. In the same way that database updates were interpreted as *changes in history*, these time paradoxes are interpreted as *the problems of changing history* or *how changes in history affect other times*.

We classify these time paradoxes and, in order to detect their occurrences, the notions of *temporal and syntactical dependences* of a rule are studied. These notions are then used to develop algorithms that perform the detection of time paradoxes. All algorithms developed in this chapter are collected and presented in the Appendix A.

5.1 Active Databases

Valid-time databases as presented in the previous chapter are passive repositories of data wherein any data is, in principle, updatable; no interaction takes place between

temporal data to either generate or remove other data, or to execute a program in the database environment. To allow for this interaction we introduce *temporal rules* in the database. The addition of rules to the database does not contradict the view of the database as a model, for the rules will not be seen as a theory from which inferences can be drawn, but as a constraints that force the two-dimensional evolution of the database to occur according to the execution semantics of the rules. As a consequence, we will be providing the database with logical links whereby the existence of some data will force the insertion or deletion of some other data.

Temporal rules were first introduced by Gabbay [1987] under the paradigm of "imperative future", which was then applied and further developed in several papers [Barringer *et al.* 1989; 1991; Loucopoulos *et al.* 1990; Manning and Torsun 1989; Finger, McBrien and Owens 1991; Finger, Fisher and Owens 1993]. Gabbay's temporal rules were temporal formulae of the form

$$\Box(Condition \rightarrow Action)$$

where *Condition* is a boolean combination of pure-present and pure-past formulae and *Action* is a pure-future formula; the propositional atoms are of two kinds, controllable and environment atoms, such that environment atoms can only appear in *Condition* but not in *Action*. The intended meaning of such rules is that, whenever *Condition* holds against the past and present data in history, *Action* is executed imperatively by making it hold in the future in the database; hence the name *imperative future*.

While that semantic interpretation of temporal rules does provide the database with links between data associated to distinct time points, and despite its intuitive appeal, the restriction to the format "past and present implies future" has a few problems.

With respect to the format of *Condition*, we note that valid-time databases can also store information about the future, which is then interpreted as an expectation about what is going to happen. It is reasonable to base the execution of actions not only on the data recorded about the past and present, but also on the expectations on the future. For example, if one expects to attend a meeting abroad in a couple of weeks time, it is reasonable to book flight tickets now, which may indeed be performed automatically if the information is present in the database; if share prices are expected to fall, it is reasonable to start selling.

On the other hand, accepting any pure-future formulae as actions brings the problem of deciding which atomic information is to be inserted in the database and at which time, a problem due to the existence of indeterminate actions of the form

$A \vee B$ and FA. It may also become intractable to decide whether a set of rules containing such actions is always consistent. Moreover, to confine actions to be pure-future formulae is too restrictive, for we have the possibility of updating data associated with any time in an valid-time database.

Besides updating the database, we want to have the ability of starting a program in the database environment by means of *external actions*, *e.g.* the automatic flight booking may be one of such programs. It is clear that external actions can be performed only at the current time.

We therefore modify the format of temporal rules to cope with the problems mentioned above. We distinguish between two kinds of atomic actions. *Database actions* are atomic updates represented by negated and non-negated atoms. *External actions* are non-negated atoms associated to programs such that, whenever an atomic external action a is executed, a is inserted at the current time in the database and its associated program π_a is executed in the database environment (in a first-order database, every external predicate action is associated with a program and the predicate arguments are seen as parameters passed to the program; all variables in an external action atom must be bound to a value at execution time, so that the propositional abstraction can be extended naturally). For example, the formula $\neg employee(\text{Peter}, \text{1K}, \text{R\&D})$ is a database deletion action and $employee(\text{Mary}, \text{5K}, \text{Finance})$ is a database insertion action; the formula $pay(x, y)$ is an external action associated with a program π that takes as parameters a person name, x, and an integer y, and prints a £y payment cheque to x.

Notation: In the following, we represent Both *Condition*- and *Action*-parts of rules as FOTL-formulae, since this is the natural way to express rules in a real database. This should bring no conflict, for we define the semantics of rule execution in terms of their propositional abstraction. Also, we use the meta-level words *Condition* and *Action* to represent first-order formulae or their propositional abstractions, i.e. we may use "*Condition*" instead of "$[\![Condition^v]\!]$, for some valuation v", and similarly with *Action*.

Definition 5.1 A *temporal rule* is a formula of the form

$$Condition(x_1, \ldots, x_m) \Rightarrow Action(x_1, \ldots, x_m)$$

where *Condition* is any temporal query formula, *i.e.* a safe formula, with free variables $x_1, \ldots, x_m$, and $Action(x_1, \ldots, x_m)$ is a *deterministic action*, which is defined as follows in two steps; first define *update actions*:

- every database action is an update action;

- if A and B are update actions, so are $\bigcirc A$, $\bullet A$ and $A \wedge B$;

then define *deterministic action*:

- every update action and every external action is a deterministic action;
- if A and B are deterministic actions, so is $A \wedge B$.

□

The double arrow ($\Rightarrow$) was used instead the single arrow ($\rightarrow$) for we are going to present two semantical interpretations of temporal rules $Condition \Rightarrow Action$, namely the execution semantics and the valid-time interpretation of rules. In none of them $\Rightarrow$ is identical to $\rightarrow$. As a consequence, both *Condition* and *Action* are safe formulae belonging to the FOTL language defined in Section 4.1, but not the rule itself.

Note that all free variables occurring in the *Action* part of the rule must occur in the *Condition* part. Since *Condition* is a safe formula, this guarantees that there will be only a finite number of actions to be executed and all its arguments will then be bound to a value.

Due to this new format of temporal rule in Definition 5.1, the intuitive notion of "past and present implies future" does not hold any more over the valid-time flow; however, it will be recovered over the transaction-time flow of time by the two-dimensional execution semantics of temporal rules given below.

Definition 5.2 (Execution Semantics) Let $\mathcal{M}$ be a two-dimensional model representing the evolution of an active valid-time database containing the set of temporal rules:

$$Condition_i \Rightarrow Action_i$$

Consider the finite set

$$act(t) = \{[\![Action_i^v]\!] \mid \text{there exists } i, v \text{ such that } \mathcal{M}, t, t \models [\![Condition_i^v]\!]\}$$

of actions fired at the diagonal point t. The *semantics of rule execution* says that if $act(t)$ is satisfiable, then $\overline{\bigcirc}(\bigwedge act(t))$ must hold at the next transaction time, *i.e.*

$$\mathcal{M}, t, t \models Condition_i \quad \text{implies} \quad \mathcal{M}, t, t+1 \models Action_i$$

□

Note that this semantics is equivalent to reading the temporal rule as the two-dimensional formula $(\delta \wedge Condition_i) \rightarrow \overline{\bigcirc} Action_i$ holding over $\mathcal{M}$, which satisfies the format "present implies future" over the transaction-time flow. The elements of $act(t)$ are called *executed actions at time t*. For every external atomic action in $act(t)$ its associated program is executed against the database environment. Rules are always executed at present time, *i.e.* on the diagonal of the two-dimensional plane; in this sense, those programs will always be executed at some current time.

If the set $act(t)$ is unsatisfiable the situation is undefined. Typically this would mean that the database transaction that has caused the generation of the invalid state will be rolled back so as to restore a satisfiable state of the database. Note that since we have deterministic actions, the unsatisfiability check can be done efficiently. The treatment of transactions remains outside the scope of this work.

Definition 5.3 (Active Database) An *active valid-time database* is a valid-time database enhanced with a finite set of temporal rules such that, if $\mathcal{M}$ is a two-dimensional representation of the evolution of the database, then $\mathcal{M}$ satisfies the execution semantics of Definition 5.2. □

This definition accommodates the view of the database as a model with the presence of rules as part of the database.

Concerning the ways an active database can interact with its environment, the presence of rules adds a bidirectionality to that interaction that does not exist in non-active databases. Both rules and updates have an effect on the evolution of the database, each taking part as one side of a two-way interaction between the database and its environment:

- Environment → Database: the environment acts upon the database by updating it.
- Database → Environment: the rules react to the data in the database, changing the data in it and, possibly, executing a program in the environment.

Example 5.1 Suppose that we add the following rule to the database of Example 4.2, so that an employee is to be payed every month the amount corresponding to the previous month salary,

$$\bullet employee(Person, Salary, Dept) \Rightarrow pay(Person, Salary).$$

Action $pay(Person, Salary)$ is associated to a program that prints a cheque of amount $Salary$ to $Person$. Suppose we have the following information in the database at transaction time $Apr93$

- $[Jan90, Dec92] : [\![employee(\text{Peter}, 1\text{K}, \text{R\&D})]\!]$
- $[Jan93, Apr93] : [\![employee(\text{Peter}, 2\text{K}, \text{Marketing})]\!]$
- $[Jan90, Dec92] : [\![employee(\text{Paul}, 1\text{K}, \text{R\&D})]\!]$
- $[Sep91, Apr93] : [\![employee(\text{Mary}, 3\text{K}, \text{Finance})]\!]$

If that information had been monthly added to the database since $Jan90$, due to the execution of the rules the following would also be in the database.

- $[Feb90, Jan93] : [\![pay(\text{Peter}, 1\text{K})]\!]$
- $[Feb93, Apr93] : [\![pay(\text{Peter}, 2\text{K})]\!]$
- $[Feb90, Jan93] : [\![pay(\text{Paul}, 1\text{K})]\!]$
- $[Oct91, Apr93] : [\![pay(\text{Mary}, 3\text{K})]\!]$

□

5.2 The Valid-time Interpretation of Rules

Suppose we have in the database only rules of the form "past implies present" and that the past is never changed by an update so that, once a rule is executed in the database, both the condition part that holds in the database (we call it the rule's *support* in the database) and the executed action that is consequently recorded in the database remains forever in the database and is never changed.

In this scenario, one may try to confront the rules of an active valid-time database against the database state, *i.e.* testing for the execution of rules at every valid-time in that state. If one does such a confrontation expecting to find that, at every valid-time, the support and the recorded actions of a rule must either both hold or fail, we say one is using a valid-time interpretation of the rules.

In the presence of arbitrary updates in the past or with rules with a more liberal format, like that we have adopted in the previous section, the valid-time interpretation of rules may not hold. This (perhaps intuitive) static view of rules does not follow, in the general case, directly from the dynamic semantics of rule execution and

it may indeed become invalid due to the occurrence of database updates, generating a *time paradox*. This section discusses the kinds of time paradoxes that may appear from the conflict between execution semantics and the valid-time interpretation in the presence of generic updates.

We can express these ideas more precisely. In a *valid-time interpretation of rules* every rule support holding at a database state should imply that its corresponding action was executed and recorded in the database; furthermore, to avoid actions being executed without any support, in a kind of "spontaneous generation of actions", a recorded action should hold in the database only if accompanied by its corresponding support; that would be equivalent to reading the "$\Rightarrow$" symbol in rules as "$\leftrightarrow$" and not simply as "$\rightarrow$". In other words, under the valid-time interpretation, the rule

$$Condition \Rightarrow Action$$

is understood as the formula

$$\Box(Condition \leftrightarrow Action)$$

holding over the database state. Such a view is the one originally presented as the declarative past and imperative future view of a temporal database; there, however, the issue of updating the past was not raised, so no conflict was generated.

There are several update generated ways of invalidating the declarative valid-time interpretation of rules, generating a time paradox. Updates, either coming from the environment or resulting from the execution of an action, can affect both the support and the record of an executed action of a rule with respect to some past time t, invalidating the valid-time interpretation; such invalidation can occur both in the case the rule was once executed in the past at t and in the case it was not, *i.e.* the support of the rule did not hold at the diagonal point t.

The importance of those time paradoxes comes from the fact that they are interpreted as being the problems that arise from changes in history. In the absence of the temporal links given by temporal rules, no such anomaly could exist. Within the framework of an active valid-time database we can therefore study the problems of changing in the past. Next we discuss and classify such time paradoxes. Their detection in a database is the subject of the remaining sections of this chapter.

5.2.1 Non-supported Actions

After the execution of a rule an update can falsify its support, leaving a recorded action in a database state in which there is no apparent justification for its existence.

Under the valid-time interpretation of rules, this is a contradiction, for *Action* holds but not *Condition*.

Formally, let $\mathcal{M}$ be a model describing the database evolution; we say that $[\![Action^v]\!]$ becomes *non-supported* at transaction time x and valid time t if it was executed in the past moment $t < x$, *i.e.* $\mathcal{M}, t, t \models [\![Condition^v]\!]$, such that until transaction time $x-1$ the support and recorded actions of the rule persist, *i.e.* for $t < y \leq x-1$, $\mathcal{M}, t, y \models [\![(Condition \wedge Action)^v]\!]$, but after an update Θ_{x-1}, $\mathcal{M}, t, x \models \neg[\![Condition^v]\!] \wedge [\![Action^v]\!]$. Typically, an update in the past is the cause for the appearance of a non-supported action in the database, but since the support of a rule is not restricted to the past only, any update can cause it.

Example 5.2 Consider the temporal active database of Example 5.1, where the action $pay(\text{Mary}, 3\text{K})$ was executed from $Sep91$ until $Apr93$. Consider the situation illustrated in Example 4.6 where, at $Apr93$, Mary's salary is increased to 5K retroactively to the beginning of the 1993 year. This would leave the database with action $pay(\text{Mary}, 3\text{K})$ non-supported from $Feb93$ until $Apr93$. In other words, at $Apr93$ there is no longer a justification in the database for having paid Mary only 3K from $Feb93$ until $Apr93$. This situation is illustrated in Figure 5.1

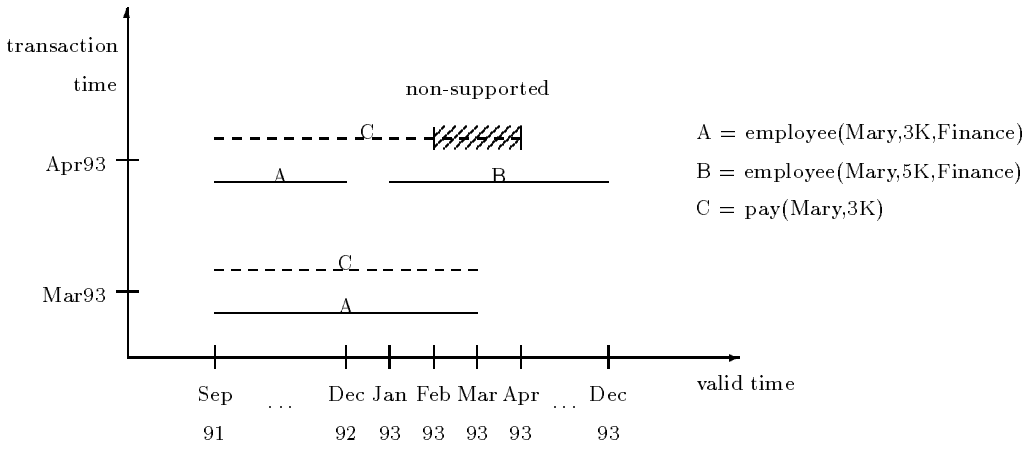


Figure 5.1 Non-supported payment

Note that the rule we are dealing with does respect the imperative future format restriction of "past implies present or future", but a non-supported actions may still occur. The only way to avoid such time paradox would be to forbid updates in the past, transforming the valid-time database into a transaction-time one. □

5.2.2 Retroactive Actions

An update, mainly one of the past, may leave the database in a state where there exists support at a past time t for an action to have been executed in the past, but the action was not executed at t. It is obvious that, if an employee is not in the database, it will not be paid a salary; salaries are only paid to employees that exist in the database *at the time of the payment*. But if an update inserts a new employee retroactively two months in the past, this employee will not have been paid two months salary because that information was not present at the time of payment. In this case, the payment of his salary is said to be a *retroactive action*.

Recall that, according to the semantics of rule execution, rules may only be executed at time points corresponding to the two-dimensional diagonal, *i.e.* at the current time, so no rule can be executed in the past. As a consequence, a retroactively fired rule and its corresponding retroactive action will never be executed.

The retroactive firing of an action can be seen as dual to the appearance of non-supported action. In the case of non-supported actions, both *Condition* and *Action* are true before the update, but after it *Condition* is falsified and *Action* still holds; in the case of retroactive actions, on the other hand, both *Condition* and *Action* are false before the update, yet *Condition* holds after the update but not *Action*.

Formally, if $\mathcal{M}$ is a model describing the evolution of an active database containing the rule $Condition \Rightarrow Action$, we say that an action or rule is *retroactively fired* at valid-time t according to transaction- time $x > t$ if $\mathcal{M}, t, t \not\models [\![Condition^v]\!]$ for some valuation v, so that the rule is not fired and $[\![Action^v]\!]$ is not executed, and this situation persists until transaction time $x-1$, *i.e.* for $t \leq y \leq x-1$, $\mathcal{M}, t, y \not\models [\![Condition^v]\!]$ and $\mathcal{M}, t, y \not\models [\![Action^v]\!]$; but after an update Θ_{x-1}, at the database state at transaction-time x, $\mathcal{M}, t, x \models [\![(Condition \wedge \neg Action)^v]\!]$. The formula $\Box(Condition \leftrightarrow Action)$ does not hold over the database state $\mathcal{M}_x$.

As in the previous case, an update in the past is typically the cause for its appearance, however since the support of a rule is not restricted to the past, any update can in principle cause it.

Example 5.3 Continuing Example 5.2 on Mary's retroactive salary salary increase from 3K to 5K, we see that the action $pay(\text{Mary}, 5\text{K})$ was retroactively fired from $Feb93$ until $Apr93$, *i.e.* it has a justification in the current state of the database for not having been executed, but the database state reflects that the execution never happened. This situation is illustrated in Figure 5.2

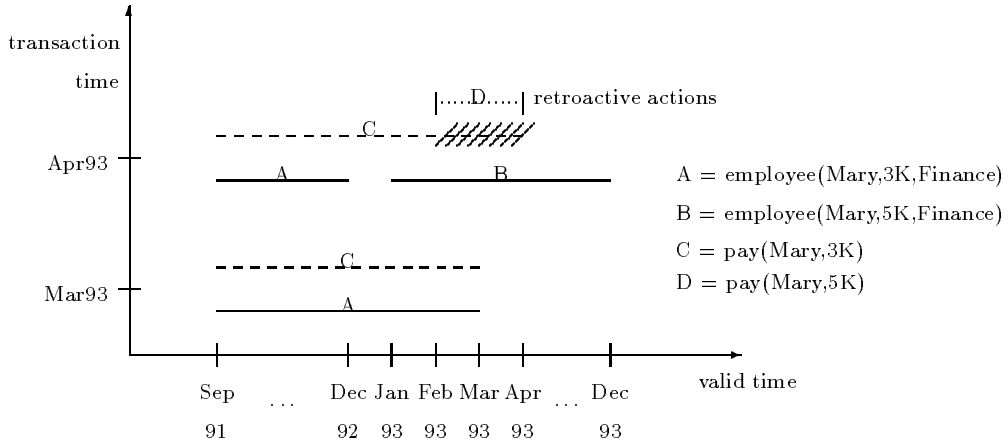


Figure 5.2 Retroactive firing of payment action

Note that there is no obvious "corrective action" to the time paradox generated by a retroactively fired action. In Mary's case, the payment of the outstanding amount may depend on the company's policy. The payment may be made through an extra cheque at the current month, or its value may be added to next month payment, or it may be converted into shares of the company, etc.

The simple correction of the violation of the valid-time interpretation is no solution to the time paradox, because, in this case, it would mean adding to the database the payment of a $5K$ cheque to Mary, which never occurred, or the deletion of her new salary, undoing her retroactive increase for the sake of a paradox-free database; clearly, those are not adequate corrective actions. The situation is not better if we try to "correct" non-supported actions. The specification of such corrective action, if done formally so as to be executed automatically, may have to take into account the dynamic nature of retroactive actions and therefore it would require the expressivity of two-dimensional temporal logic. We do not explore these "corrective actions" in this work, concentrating only on the detection of the occurrence of problematic situations. □

Examples 5.2 and 5.3 show the simultaneous occurrence of non-supported and retroactive actions due to a modification update. When this happens, we call them *connected actions*. Note, however, that there can exist non-supported actions without a connected retroactive one, *e.g.* the removal of all past records of an employee leaving all past payments non-supported. There can also exist retroactive actions without a connected non-supported one, *e.g.* the retroactive hiring of an employee, retroactively firing payments that never occurred.

5.2.3 Rule Violation and Faked Execution

After a rule was triggered and executed at valid time t, an update may remove the recorded actions of the rule, leaving the database in a state that, at t, the support of a rule is in the database but not its corresponding recorded actions. Therefore, it is said that the execution of the rule was violated.

Let $\mathcal{M}$ be a two dimensional model representing the evolution of an active valid-time database and let $Condition \Rightarrow Action$ be a rule in it. We say that there is a *rule violation* in $\mathcal{M}$ at valid time t and transaction time x if:

$$\mathcal{M}, t, x \models Condition \wedge \neg Action \wedge \overline{S}\,(\delta \wedge Condition, Condition \wedge Action)$$

It excludes the case where both the recorded actions and the support of a rule are simultaneously deleted, for then there is no invalidation of the valid-time interpretation.

The dual of rule violation occurs when an update in the past inserts in the database the recorded actions of a rule that was not fired at that past time, therefore faking its execution.

We say that a *faked execution* is introduced in $\mathcal{M}$ at valid time t and transaction time x if :

$$\mathcal{M}, t, x \models Action \wedge \overline{S}\,(\delta \wedge \neg Condition, \neg Action).$$

Note that this definition excludes the case where the action was executed, then a later update removed both its recorded actions and support, and then only the recorded actions are restored to the database; in this case, the execution is not considered faked because it actually happened, although it does violate the valid-time interpretation. The execution is considered faked even if the support of the rule is simultaneously inserted in the database with its recorded actions, even though there is no violation of the valid-time interpretation in this case; programs associated with an external action included in the faked recorded actions will not be executed, for programs are executed only at a present time (note that it follows from the definition that $t < x$), so the execution is still considered faked.

Faked actions are not considered a serious problem that deserves having its occurrences always detected. In fact, a faked execution may be even part of the "corrective action" taken in the case of a retroactively fired rule, in which case the faked action is not seen as a paradox any more.

5.2.4 Summary

Table 5.1 contains a summary of the possible effects of temporal updates in active valid-time databases. The first four rows present the update generated time paradoxes arising from conflicts between the execution semantics and the valid-time interpretation of rules. The other three rows present the cases where no invalidation of the valid-time interpretation occurs, namely when neither support nor recorded actions of the rule is changed, and when both support and recorded actions are simultaneously removed from the database, therefore keeping the validity of the valid-time interpretation.

The fact that an action was executed or not at a certain time t cannot be detected by looking at the state of a one-dimensional valid-time database. If the support of a rule is in the database at time t but not its recorded actions, it may either be the case that it was there at execution time, but an update later removed its recorded actions, or it may be the case that the support did not hold at t, but was later introduced by an update. The detection of a past execution is, in fact, two-dimensional and cannot be extracted from the database state.

	Executed	*Support*	*Recorded Action*
Non-supported Action	Yes	False	True
Retroactive Action	No	True	False
Rule Violation	Yes	True	False
Faked Execution	No	True/False	True
No Change	Yes	True	True
	No	False	False
Simultaneous Deletion	Yes	False	False

Table 5.1 Effects of temporal updates in active databases

In the following we concentrate on a method for detecting the occurrence of such time paradoxes.

5.3 Syntactical and Temporal Dependences

We describe here a method for detecting the appearance of non-supported actions in the database and we show how this method can be used to detect their connected retroactive actions. The method produces, as a side effect, a way of detecting rule violation, but we do not attempt to detect faked executions. The detection of loss of support is particularly interesting in the case of non-supported external actions, for

then a program has no justification in the database for its past execution against the database environment and, since we cannot change the past state of the environment, those actions are of particular interest for detection. In the following, whenever we refer to a non-supported or retroactive action, unless otherwise specified, we mean an external action.

The naive method for detecting loss of support consists of rechecking the database for rule support of executed actions at every past time point after every update. Of course, this solution just uses brute force and is computationally very expensive; therefore it is unacceptable.

The first thing deserving notice is that it is not necessary to check every rule at every time point in the past if we keep a log of executed rules and their execution times; we postpone an exact description of this log until later, but note that this log may increase indefinitely, so we concentrate on the detection of loss of support in a "recent past", *i.e.* we may fix, *a priori*, the time interval we will be searching in the past. The log allows us to recheck only the rules that were once executed.

However, even with the log there are still too many checks to be done, for an update usually does not affect all the data at all times in the database. Ideally, we should only recheck the support of executed rules whose support was affected by an update. To deal with this idea of "affected support" we introduce the notions of *syntactic dependence* and *temporal dependence* of a query.

The syntactic dependence of a first-order formula consists of the set of the predicate symbols occuring in the formula. The *positive syntactic dependence* of a formula is the set of predicate symbols occurring within the scope of an even number of "$\neg$" symbols; the *negative syntactic dependence* of a formula is the set of predicate symbols occurring within the scope of an odd number of "$\neg$" symbols. The positive/negative dependences of a rule are those of its *Condition*-part.

The temporal dependences of a formula A at a valid time t consist of sets of time points at which the truth or falsity of atomic formulae in the database gives support to the truth of A at t and, as in the syntactical case, there are positive and negative temporal dependences. It follows that the temporal dependences of a formula actually depends on the database state, *i.e.* it relies on the semantics of the formula. In order to formally define the temporal semantics, we make use of an extended representation of temporally labelled formulae and we define the semantics of such labelled formulae where, if t is a time point, d_+ and d_- are sets of time points and A is a temporal formula, $(t, d_+, d_-) : A$ is a well formed temporally labelled formula. Let x be the current time and let $\mathcal{M}_x = (\mathbb{Z}, <, h)$ be a temporal

model representing the database state at x; the expression

$$\mathcal{M}_x \models (t, d_+, d_-) : A$$

means that the formula A is true in $\mathcal{M}_x$ at time t with positive temporal dependence d_+ and negative temporal dependence d_-. Table 5.2 contains the extended definition of the semantics of temporally labelled formulae with temporal dependences.

Note that $\mathcal{M}_x \not\models (t, d_+, d_-) : A$ does not imply $\mathcal{M}_x \models (t, d_+, d_-) : \neg A$; the former means that A is not true at t with temporal dependences d_+ and d_-, but it may well be true at t with other sets of dependences; the latter means that $\neg A$ is true at t with the dependences d_+ and d_-. Table 5.2 therefore treats separately each case of negation. Note that it is also possible to have several distinct temporal dependences for the same formula at the same time, *e.g.* if $s < u < t$ and q holds at s and u then both $\mathcal{M}_x \models (t, \{s\}, \varnothing) : Pq$ and $\mathcal{M}_x \models (t, \{u\}, \varnothing) : Pq$. As in the syntactic case, the positive/negative dependences of a rule at time t are those of its *Condition*-part.

Lemma 5.1 *There exist temporal dependence sets d_+ and d_- such that $\mathcal{M}_x \models (t, d_+, d_-) : A$ if and only if $\mathcal{M}_x, t \models A$.*

Proof The proof is by induction on Table 5.2; the "if and only if" is part of the induction hypothesis. For the basic cases, note that $p \in h(t)$ iff $\mathcal{M}_x, t \models p$ iff $\mathcal{M}_x \models (t, \{t\}, \varnothing) : p$ and $p \notin h(t)$ iff $\mathcal{M}_x, t \models \neg p$ iff $\mathcal{M}_x \models (t, \varnothing, \{t\}) : \neg p$. The non-negated cases in Table 5.2 are straightforward to prove and are therefore omitted; double negation and negation of conjunction are also straightforward and omitted.

The interesting parts of Table 5.2 are those concerning the negation of the temporal operators over $\mathbb{Z}$. We discuss here the case for the S-operator; for the U-operator the situation is analogous. Recall the semantics of $S(A, B)$ where it holds at a point t iff

(a) A holds somewhere to the past of t, at s; and

(b) B holds at all points u between s and t.

The negation of the formula $S(A, B)$ is satisfied if either of those cases is not. The first one is not satisfied if there is no such s at the past where A holds, so $H\neg A$ holds at t; by induction hypothesis, $\mathcal{M}_x \models (s, d^s_+, d^s_+) : \neg A$ for all $s < t$ and by Table 5.2, $\mathcal{M}_x \models (t, \bigcup_{s<t} d^s_+, \bigcup_{s<t} d^s_+) : \neg S(A, B)$. The second one fails to hold over an integer-like flow of time if, going towards the past, we reach $\neg B$ before we reach A; over a $\mathbb{Z}$-like flow of time, this means that $\neg B \wedge \neg A$ is satisfied in the past and since

$\mathcal{M}_x \models (t, \{t\}, \varnothing) : p$	iff	$p \in h(t)$
$\mathcal{M}_x \models (t, \varnothing, \{t\}) : \neg p$	iff	$p \notin h(t)$
$\mathcal{M}_x \models (t, d_+, d_-) : \neg\neg A$	iff	$\mathcal{M}_x \models (t, d_+, d_-) : A$.
$\mathcal{M}_x \models (t, d_+, d_-) : A \wedge B$	iff	$\mathcal{M}_x \models (t, d'_+, d'_-) : A$ and $\mathcal{M}_x \models (t, d''_+, d''_-) : B$ where $d_+ = d'_+ \cup d''_+$ and $d_- = d'_- \cup d''_-$.
$\mathcal{M}_x \models (t, d_+, d_-) : \neg(A \wedge B)$	iff	$\mathcal{M}_x \models (t, d_+, d_-) : \neg A$ or $\mathcal{M}_x \models (t, d_+, d_-) : \neg B$
$\mathcal{M}_x \models (t, d_+, d_-) : S(A, B)$	iff	there exists $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : A$, and for all u, $s < u < t$, $\mathcal{M}_x \models (u, d^u_+, d^u_-) : B$, where $d_+ = \bigcup_{s \le v < t} d^v_+$ and $d_- = \bigcup_{s \le v < t} d^v_-$.
$\mathcal{M}_x \models (t, d_+, d_-) : \neg S(A, B)$	iff	for all $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg A$, where $d_+ = \bigcup_{s < t} d^s_+$ and $d_- = \bigcup_{s < t} d^s_-$; or there exists $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg B \wedge \neg A$, and for all u, $s < u < t$, $\mathcal{M}_x \models (s, d^u_+, d^u_-) : \neg A$ where $d_+ = \bigcup_{s \le v < t} d^v_+$ and $d_- = \bigcup_{s \le v < t} d^v_-$.
$\mathcal{M}_x \models (t, d_+, d_-) : U(A, B)$	iff	there exists $s > t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : A$, and for all u, $t < u < s$, $\mathcal{M}_x \models (u, d^u_+, d^u_-) : B$, where $d_+ = \bigcup_{t < u \le s} d^u_+$ and $d_- = \bigcup_{t < u \le s} d^u_-$.
$\mathcal{M}_x \models (t, d_+, d_-) : \neg U(A, B)$	iff	for all $s > t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg A$ where $d_+ = \bigcup_{t < s} d^s_+$ and $d_- = \bigcup_{t < s} d^s_-$; or there exists $s > t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg B \wedge \neg A$ and for all u, $t < u < s$, $\mathcal{M}_x \models (s, d^u_+, d^u_-) : \neg A$ where $d_+ = \bigcup_{t \le v < s} d^v_+$ and $d_- = \bigcup_{t \le v < s} d^v_-$.

Table 5.2 Temporal dependences

then $\neg A$ holds, which can be expressed as the formula $S(\neg B \wedge \neg A, \neg A)$ holding at t; therefore, by induction hypothesis, there exists $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg B \wedge \neg A$, and for all u, $s < u < t$, $\mathcal{M}_x \models (s, d^u_+, d^u_-) : \neg A$, and by Table 5.2, $\mathcal{M}_x \models (t, \bigcup_{s \leq v < t} d^v_+, \bigcup_{s \leq v < t} d^v_-) : \neg S(A, B)$. For dense flows of time and for flows that allow "gaps"[1], there are other possibilities to falsify the second case which we need not take into account here. On the other hand, if $\mathcal{M}_x \models (t, d_+, d_-) : \neg S(A, B)$, then by Table 5.2 either of the above two cases is unsatisfied, so the induction hypothesis gives us $\mathcal{M}_x, t \models \neg S(A, B)$. This finishes the proof. □

When there is a generic update in the database, several time points are affected. Let Θ_x be an update determined by the pair (θ_+, θ_-). The set of time points positively affected by the update, $Aff_+(\Theta_x)$, is defined as

$$Aff_+(\Theta_x) = \{t \mid t : q \in \theta_+\}$$

and the set of time points negatively affected by the update, $Aff_-(\Theta_x)$, as

$$Aff_-(\Theta_x) = \{t \mid t : q \in \theta_-\}.$$

The support of formulae is preserved after an update under the following case.

Lemma 5.2 *For every transaction time x and every valid time t and every formula A such that $\mathcal{M}_{x-1} \models (t, d_+, d_-) : A$, if $Aff_+(\Theta_{x-1}) \cap d_- = Aff_-(\Theta_{x-1}) \cap d_+ = \varnothing$ then $\mathcal{M}_x \models (t, d_+, d_-) : A$*

Proof By induction on Table 5.2. For the base cases, if $A = q$ then $d_+ = \{t\}$ and $d_- = \varnothing$; it follows that $t : q \notin \theta_-$, so by update semantics we have $\mathcal{M}_x, t \models q$ and $\mathcal{M}_x \models (t, d_+, d_-) : q$. If $A = \neg q$ then $d_- = \{t\}$ and $d_+ = \varnothing$; it follows that $t : q \notin \theta_+$, so $\mathcal{M}_x \models (t, d_+, d_-) : \neg q$. This finishes the base cases.

For the inductive cases, we only examine the cases involving the temporal operator S; the other cases are either analogous or straightforward. If $\mathcal{M}_{x-1} \models (t, d_+, d_-) : S(A, B)$ then there exists $s < t$, $\mathcal{M}_{x-1} \models (s, d^s_+, d^s_-) : A$, and for all u, $s < u < t$, $\mathcal{M}_{x-1} \models (u, d^u_+, d^u_-) : A$, where $d_+ = \bigcup_{s \leq v < t} d^v_+$ and $d_- = \bigcup_{s \leq v < t} d^v_-$. By induction hypothesis, there exists $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : A$, and for all u, $s < u < t$, $\mathcal{M}_x \models (u, d^u_+, d^u_-) : A$, so $\mathcal{M}_x \models (t, d_+, d_-) : S(A, B)$.

If $\mathcal{M}_{x-1} \models (t, d_+, d_-) : \neg S(A, B)$ we have to examine two cases. Suppose for all $s < t$, $\mathcal{M}_{x-1} \models (s, d^s_+, d^s_-) : \neg A$, where $d_+ = \bigcup_{s < t} d^s_+$ and $d_- = \bigcup_{s < t} d^s_-$; in this case, by induction hypothesis, for all $s < t$, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg A$ and $\mathcal{M}_x \models (t, d_+, d_-) :$

[1] A flow contains a gap if it contains an infinite ascending/descending sequence but does not contain the least upper/greatest lower bound of such sequence.

$\neg S(A,B)$. For the second case, suppose that there exists $s < t$, $\mathcal{M}_{x-1} \models (s, d^s_+, d^s_-) : \neg B$, and for all u, $s < u < t$, $\mathcal{M}_{x-1} \models (u, d^u_+, d^u_-) : \neg A$ where $d_+ = \bigcup_{s \leq v < t} d^v_+$ and $d_- = \bigcup_{s \leq v < t} d^v_-$. Then, by induction hypothesis, $\mathcal{M}_x \models (s, d^s_+, d^s_-) : \neg B$, and for all u, $s < u < t$, $\mathcal{M}_x \models (u, d^u_+, d^u_-) : \neg A$; it follows that $\mathcal{M}_x \models (t, d_+, d_-) : \neg S(A,B)$, which finishes the proof. □

By combining temporal dependences and temporal affectedness we get the following necessary condition for an action to become non-supported.

Theorem 5.1 (Non-supported actions) *Let $\mathcal{M}$ describe the evolution of an active temporal database containing the rule Condition $\Rightarrow$ Action such that $\mathcal{M}_{x-1} \models (t, d_+, d_-) : Condition$. A necessary condition for an executed Action to become non-supported at transaction time x and valid time t is*

$$[Aff_+(\Theta_{x-1}) \cap d_-] \cup [Aff_-(\Theta_{x-1}) \cap d_+] \neq \varnothing$$

Proof Suppose $\mathcal{M}_{x-1} \models (t, d_+, d_-) : Condition$ so, by Lemma 5.1, $\mathcal{M}_{x-1}, t \models Condition$. If we assume that $Aff_+(\Theta_{x-1}) \cap d_- = Aff_-(\Theta_{x-1}) \cap d_+ = \varnothing$, by Lemma 5.2 it follows that $\mathcal{M}_x \models (t, d_+, d_-) : Condition$ and, by Lemma 5.1, $\mathcal{M}_x, t \models Condition$, which contradicts the fact that *Action* becomes non-supported at transaction time x and valid time t. □

A similar result can be obtained for the syntactic dependences on the first-order view of a database. Let Θ_x be an update determined by (θ_+, θ_-). Let $Pred(\theta_+)$ be the set of predicate symbols whose $[\![]\!]$-abstraction occurs in θ_+, and similarly for $Pred(\theta_-)$ with respect to θ_-.

Theorem 5.2 (Syntactic Dependencies) *Let $\mathcal{M}$ describe the evolution of an active temporal database containing the rule Condition $\Rightarrow$ Action such that s_+ and s_- are, respectively, its positive and negative syntactic dependences and $\mathcal{M}_{x-1}, t \models Condition$. A necessary condition for an executed Action to become non-supported at transaction time x and valid time t such that Θ_x is an update determined by (θ_+, θ_-) is*

$$[Pred(\theta_+) \cap s_-] \cup [Pred(\theta_-) \cap s_+] \neq \varnothing$$

Proof Suppose $\mathcal{M}_{x-1}, t \models Condition^v$ and $\mathcal{M}_x, t \models \neg Condition^v$ for some v. A simple induction on the structure of *Condition* shows us that $[Pred(\theta_+) \cap s_-] \cup [Pred(\theta_-) \cap s_+] \neq \varnothing$. For the base cases, if $Condition = p(x)$ then $s_+ = \{p\}$ $s_- = \varnothing$; then *Condition* can only be falsified if $t : [\![p(x)^v]\!]$ is deleted, so $p \in Pred(\theta_-)$. If $Condition = \neg p(x)$ then $s_- = \{p\}$ and $s_+ = \varnothing$; then *Condition* can only be

falsified if $t : [\![p(x)^v]\!]$ is inserted, so $p \in Pred(\theta_+)$. The inductive cases are all straightforwardly proved and we omit the details. □

Since the notion of syntactical dependence is primarily a first-order one, it is reasonable to ask how the definition of temporal dependences translates from the propositional abstraction into the first-order case. For that, we combine the definition of temporal dependences from Table 5.2 and the propositional abstraction from Section 4.2; recall that $R_{\mathcal{D}}$ is the set of all relevant domain elements of the database $\mathcal{D}$; furthermore we apply the three-place labels to safe first-order temporal formulae. The quantifier free cases are basically those of Table 5.2; for the quantified cases we obtain:

$\mathcal{D}, v \models (t, d_+, d_-) : \exists x A$ iff there exists v' an x-variant of v such that $v'(x) \in R_{\mathcal{D}}$ and $\mathcal{D}, v' \models (t, d_+, d_-) : A$.

$\mathcal{D}, v \models (t, d_+, d_-) : \neg\exists x A$ iff for every v' an$x$-variant of v such that $v'(x) \in R_{\mathcal{D}}$, $\mathcal{D}, v' \models (t, d_+^{v'(x)}, d_-^{v'(x)}) : \neg A$, where $d_+ = \bigcup_{c \in R_{\mathcal{D}}} d_+^c$ and $d_- = \bigcup_{c \in R_{\mathcal{D}}} d_-^c$.

It follows from the safeness of formula A that both Lemmas 5.1 and 5.2 and Theorem 5.1 are extended to the first-order case with the definition above complementing Table 5.2. Note that a formula of the form $\exists x A$ may have several distinct temporal dependences.

We now examine the feasibility of computing the temporal dependences during query evaluation. Table 5.2 can be seen as a reasonable means for actually computing the temporal dependences during a query evaluation; in fact, real databases containing large tables demand better optimisations to achieve acceptable response times for queries, but for the sake of showing the feasibility of computing temporal dependences for quantifier free formulae, we consider Table 5.2 satisfactory; the issue of incorporating the computation of temporal dependences to the optimisation of queries is outside the scope of this work.

The extension dealing with quantifiers, however, does present us with a computational problem. Although the positive version of $\exists x A$ does not pose any problem, the negative case of $\neg\exists x A$ seems to demand that the temporal dependences for $\neg A(x)$ be calculated for each element of $R_{\mathcal{D}}$, which places a great computational burden; note that it leads to considering domain elements that are not even relevant to A. One possible improvement is to consider only the domain elements that are relevant to A, but this would still place a considerable burden on the system. We propose here a more straightforward computation of the dependencies in that case;

such a simplification will not affect the correctness of Algorithm 5.5 for the detection of non-supported actions.

Consider the labelled formula $(t, d_+, d_-) : \neg\exists x A$. The computation of (d_+, d_-) is done in the following (syntactical) way. We start with $d_+ = d_- = \varnothing$ and let $q(x)$ represent an atom containing the variable x. Then:

- if there is an atom $q(x)$ occurring positively in A, but not within the scope of any temporal operator, then $d_- := \{t\}$;
- if there is an atom $q(x)$ occurring negatively in A, but not within the scope of any temporal operator, then $d_+ := \{t\}$;
- if there is an atom $q(x)$ in A occurring positively inside the scope of a past operator, but not within the scope of any future operator, then $d_- := d_- \cup \{s \mid s < t\}$;
- if there is an atom $q(x)$ in A occurring negatively inside the scope of a past operator, but not within the scope of any future operator, then $d_+ := d_+ \cup \{s \mid s < t\}$;
- if there is an atom $q(x)$ in A occurring positively inside the scope of a future operator, but not within the scope of any past operator, then $d_- := d_- \cup \{s \mid s > t\}$;
- if there is an atom $q(x)$ in A occurring negatively inside the scope of a future operator, but not within the scope of any past operator, then $d_+ := d_+ \cup \{s \mid s > t\}$;
- if there is an atom $q(x)$ in A occurring positively inside the scope of both a future and a past operator, then $d_- := \mathbb{Z}$;
- if there is an atom $q(x)$ in A occurring negatively inside the scope of both a future and a past operator, then $d_+ := \mathbb{Z}$.

This leaves us with the following result.

Proposition 5.1 *There is a polynomial time algorithm that calculates the temporal dependences of a formula against a temporal database state.*

We conclude that it is effective to compute the temporal dependences of queries, which will then allow us to detect some of the time paradoxes described in Section 5.2.

5.4 Detection of Time Paradoxes

The results of the previous section allow us to discuss ways of detecting some of the time paradoxes arising from the conflicts between the execution semantic of temporal rules and the valid-time interpretation of rules. We concentrate basically on the algorithm for the detection of non-supported external actions, *i.e.* actions that caused a program to be executed against the environment; eventually other kinds of time paradoxes will be detectable following this path. The presentation of the algorithms will be done in a pseudo-programming language. All algorithms are collected and presented in Appendix A.

We represent the data structures by means of Prolog-style structures, *i.e.* functional terms of first-order logic. We start by indicating the representation of temporal dependences as a list of pairs of integer numbers

$$[(s_1, e_1), \ldots, (s_n, e_n)]$$

such that, for every i, $1 \leq i \leq n$, s_i and e_i are integer numbers, $s_i \leq e_i$ and $s_{i+1} > e_i + 1$. The pairs (s_i, e_i) intends to represent the interval $\{x \mid s_i \leq x \leq e_i\}$ and the list is supposed to represent the disjoint union of those intervals; eventually, s_1 may be equal to "*", where the pair $(*, e_1)$ represents the set $\{x \mid x \leq e_1\}$, and similarly e_n may be equal to "*" so that $(s_n, *)$ represents the set $\{x \mid x \geq s_n\}$. According to that definition, the list $[(*, 5), (6, 9), (12, 12)]$ is not acceptable as a representation of temporal dependences because $s_2 = 6$ is not greater than $e_1 + 1 = 5 + 1$. If we coalesce the two initial pairs, generating the list $[(*, 9), (12, 12)]$, we obtain an acceptable representation. This process of generating acceptable representations from generic lists of pairs can be done automatically; a description of such a *coalesce function* can be found in [McBrien 1992], together with a description of algorithms on how to insert or delete time points to this representation of sets of time points.

We assume that every rule in the active database has a unique identification. This information will be used to indicate which rules will have to be rechecked for the purpose of confirming its support.

Two-auxiliary tables are defined in the form of prolog predicates. The first table is a static one, *i.e.* it can be generated taking as input only the set of rules in the database and does not depend on the contents of the database at any time. The *table of syntactical dependences* has the form:

$$syntactical\_dependences(Pred\_Name, Pos\_dep\_list, Neg\_dep\_list)$$

where *Pred_Name* is the name of a predicate in the database, *Pos_dep_list* is a list of rule identifiers such that each correspondent rule has *Pred_Name* as a positive syntactical dependency; similarly for *Neg_dep_list* with respect to negative syntactical dependency. In order to construct such a table we do the following. For each rule, we construct the parsing tree for its *Condition*-part; predicate names will be at the leaves of the tree. We then assign either '+' or '-' to every node in the tree in the following way. The root node receives '+'. If a node contains the ¬-symbol, its children receive the opposite sign as the node itself received; otherwise, the children receive the same sign as the father. For each predicate name on a leaf we include the rule identifier on its list of positive dependences if it is assigned a '+'; otherwise we include it in the list of negative dependences. Figure 5.3 shows the parsing tree for the formula $\exists x \exists y \neg(\neg p(x) \wedge (p(y) \wedge q(x)))$, in which p occurs both positively and negatively, and q occurs negatively.

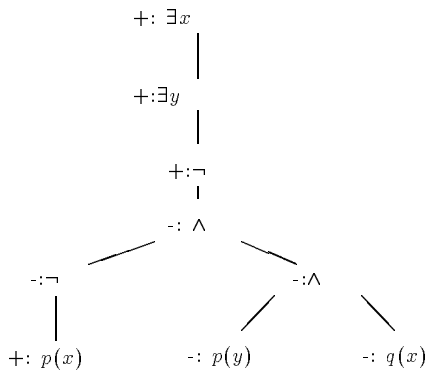


Figure 5.3 Parsing tree for $\exists x \exists y \neg(\neg p(x) \wedge (p(y) \wedge q(x)))$

Pred_Name is supposed to be the key of this table, so there are no two entries at the table with the same value for *Pred_Name*. For convenience, we use the following two functions to access the *syntactical_dependences*. All algorithms are presented in an informal pseudo-programming language, in the fashion algorithms are presented in basic books of relational database theory such as [Maier 1983] and [Ullman 1988]. An SQL-like notation is used to access the tables.

Algorithm 5.1 Positive syntactical dependences

```
Input:  a predicate name
Output:  a list of rule identifiers
POSSYNT(PredName)
BEGIN
Select Pos_dep_list from syntactical_dependences
```

```
   where Pred_Name = PredName;
return Pos_dep_list;
END
```

Algorithm 5.2 Negative syntactical dependences

```
Input:  a predicate name
Output:  a list of rule identifiers
NEGSYNT(PredName)
BEGIN
Select Neg_dep_list from syntactical_dependences
   where Pred_Name = PredName;
return Neg_dep_list;
END
```

The second table is a dynamic one, *i.e.* it is constructed as the database runs, and contains information about the external actions that were executed in the past by the system. The table $executed\_action(Action, Rule\_id, Parms, Time, d_+, d_-)$ consists of the following:

Action The action name and the parameters passed to the external action at execution time are stored in the format of a Prolog term, *e.g.* $pay(\text{Peter}, 50)$.

Rule_id The identifier of the rule whose firing caused the action to be executed.

Parms The list of parameters $[x_1, \ldots, x_m]$ that the condition part of the rule, $Condition(x_1, \ldots, x_m)$, is true of at the time of execution.

Time The time at which the action was executed.

d_+, d_- Lists representing positive and negative temporal dependences of the rule at $Time$.

The insertion of information in the table occurs at execution time. We assume that the temporal dependences are generated together with the query evaluation according to the description for first-order formulae in the previous section. The key for this table is composed of the *Action* name and parameters, and the $Time$ of its execution. This table implements the previously cited "log of executed actions" and it is a substitute for storing all the information about the transaction dimension.

For convenience, we provide two functions to access its table. Both take as input a rule identifier and a set of time points and both return a set of rows from

the *executed_action* table such that, in the first one, the row contains the input rule identifier and a positive d_+ intersection with the input set of time points is non-empty; the second one does a similar thing with respect to d_-.

Algorithm 5.3 Positive intersection rows

```
Input:  a rule ID and a list of time points.
Output:  a set of rows from executed_action table.
POSROWS(RID, TIMES)
BEGIN
ROWS := ∅;
For every row of the table given by
   Select Action, Rule_id, Parms, Time, d+, d-
   from executed_action
   where RID = Rule_id and d+ ∩ TIMES ≠ ∅
   do    ROWS := ROWS ∪{row(Action, Rule_id, Parms, Time, d+, d-)}
END
```

Algorithm 5.4 Negative intersection rows

```
Input:  a rule ID and a list of time points.
Output:  a set of rows from executed_action table.
NEGROWS(RID, TIMES)
BEGIN
ROWS := ∅;
For every row of the table given by
   Select Action, Rule_id, Parms, Time, d+, d-
   from executed_action
   where RID = Rule_id and d- ∩ TIMES ≠ ∅
   do    ROWS := ROWS ∪{row(Action, Rule_id, Parms, Time, d+, d-)}
END
```

The detection of the appearance of non-supported actions takes as input the pair (θ_+, θ_-) that characterise an update. Each set is represented as a finite list of labelled elements of the form:

$$times : atom$$

where *times* is a Prolog-style list representing a set of time points as described previously and *atom* is a ground atomic predicate in the database. We also assume there is a function PREDNAME that takes as input an atom and returns its predicate name, and a function COND that takes as input a rule identifier and returns the condition part of that rule. We have then the following algorithm to detect the occurrence of non-supported actions.

Algorithm 5.5 Detection of non-supported actions

```
Input:  update sets θ+ and θ−.
Output:  A set of time-labelled non-supported actions

DETECT_NONSUP(θ+, θ−)
BEGIN
NONSUP := ∅;

/* The first part of the algorithm deals with insertions */
For every times : atom in θ+
BEGIN
   For every rule id RID in table NEGSYNT( PREDNAME( atom ) )
   BEGIN
      For each row(Action, RID, Parms, Time, d+, d−)
      in NEGROWS( RID, times )
      BEGIN
         If the query COND( RID ) is unsatisfied by Parms at Time
         and Time : Action holds in the database
         then
            NONSUP := NONSUP ∪{Time : Action};
            delete the row from the executed_action table.
            /* OBS */
         else if COND( RID ) is satisfied with new temporal
            dependences d'+ ≠ d+ or d'− ≠ d−,
            then modify the row in the executed_action table
            with dependences d'+ and d'−.
      END
   END
END
```

```
/* The second part of the algorithm deals with deletions */
 For every times : atom in θ_
BEGIN
   For every rule id RID in table POSSYNT( PREDNAME( atom ) )
   BEGIN
      For each row(Action, RID, Parms, Time, d+, d−)
      in POSROWS( RID, times )
      BEGIN
         If the query COND( RID ) is unsatisfied by Parms at Time
         and Time : Action holds in the database
         then
            NONSUP := NONSUP ∪{Time : Action};
            delete the row from the executed_action table.  /* OBS */
         else if COND( RID ) is satisfied with new temporal
            dependences d'+ ≠ d+ or d'− ≠ d− ,
            then modify the row in the executed_action table
            with dependences d'+ and d'− .
      END
   END
END
return( NONSUP )
END
```

The parts of the algorithm with the comments `/* OBS */` will be used later when we extend it to detect connected retroactive actions. We first prove the correctness of the algorithm as it is.

Theorem 5.3 (Correctness of Algorithm 5.5) *If an external action becomes non-supported, Algorithm 5.5 will detect it.*

Proof The Algorithm 5.5 selects a (possibly empty) subset of rules to have its *Condition*-part rechecked. This selection occurs in two stages for both insertion and deletion. The first stage consists of selecting the rules whose syntactical dependences are affected by the update; due to Theorem 5.2, no relevant rule is missed out. With this initial selection as input, a second selection is done based on the temporal dependences. In the non-quantified case, the correctness of this second selection

follows directly from Theorem 5.1; the quantified case follows from the discussion at the end of last section, where we generate a big enough interval for the temporal dependences. As a consequence, we are guaranteed to recheck every rule that loses its support after the update.

Obviously, we are assuming here that the tables *syntactical_dependences* and *executed_action* are generated and maintained correctly as described previously. In fact, the correct maintenace of the latter affects the correctness of the algorithm, as we will see next.

When the condition part is rechecked, three possibilities may occur. The first is that the rule is satisfied with the same temporal dependences as those stored in *executed_action*, in which case there is nothing to be done. The second possibility is when the query is satisfied, but with temporal dependences distinct from those stored at the table; this corresponds to the fact that the update may change the support of a rule (*i.e.* the evaluation of the condition now visits different time points) without leaving it unsatisfied; Theorem 5.1 uses the value of the temporal dependences just before the update, not at evaluation time, so in order to use it correctly to select the set of rules to recheck, we have to update table *executed_action* with the current version of d_+ and d_-, which is what the algorithm does. Finally, if the recheck fails and $Time : Action$ still holds then, by definition, *Action* has become non-supported at the current transaction time, at the valid time $Time$ when the rule was executed in the past. □

A slight alteration to Algorithm 5.5 can do better than just detect non-supported actions. In the case that the *Condition*-part of a rule is not satisfied by the parameters *Parms* in the *executed_action* table, it is possible that it is satisfied by a different set of parameters that did not satisfy it at the execution time. This characterizes the appearance of a *retroactive action* connected to the non-supported action just detected. This, however, does not guarantee that all retroactive actions are detected, just those connected ones. In order to change Algorithm 5.5 to detect all the connected retroactive actions, we add to the initialization the variable `RETRO` initially set to ∅ and that is also returned at the end. Then, in the places where the comment `/* OBS */` occurs in Algorithm 5.5, we add the following piece.

Algorithm 5.6 Detection of connected retroactive actions

```
/* OBS:A non-supported action was detected at
   row(Action, RID, Parms, Time, d+, d-)
*/
```

If the query COND(RID) is satisfied at $Time$ by $Parms' \neq Parms$
and there is no row in table $executed\_action$ such that
it contains $Action(Parms')$, RID, $Parms'$ and $Time$
then
RETRO := RETRO $\cup\{Time : Action(Parms')\}$

In the algorithm above, $Action(Parms')$ represent the action we obtain by suitably substituting the new parameters in $Action$. We could execute Algorithm 5.6 after every recheck instead, so that we may even find some retroactive actions not connected to any non-supported action. Although this would generate only correct answers, *i.e.* every action detected has been retroactively fired, there are still no guarantees that all retroactive actions would be detected. The problem of finding all the retroactive actions comes from the fact that no information is stored about the rules and parameters that were unsatisfied at execution time. This must clearly be so, for there are infinitely many ways a rule might be unsatisfied (given a countably infinite domain). Nor can we afford to recheck every rule at every time point in the past after every update. So the detection of just connected retroactive actions seems a good compromise.

A complete version of Algorithm 5.5, including two occurences of Algorithm 5.6 and the correct initialization of RETRO, is presented in the Appendix A.

Note that in Algorithm 5.5 the detection of one non-supported action is independent of all the others. We may decide that it is worthwhile to apply the detection just to a certain group of rules and with the same algorithm; that would decrease the size of both auxiliary tables and consequently the time of rechecking, making the whole process more efficient. The algorithm will then correctly select the rules from the restricted set that must be rechecked.

The propagation of loss of support is one issue not discussed yet. Suppose we have the following rule

$$\bullet^7 wednesday \Rightarrow wednesday$$

stating that if seven days ago was wednesday, then today is wednesday again. Suppose we have been executing this rule for several months and the database is populated with several *wednesday*s, when we delete the first occurrence of a *wednesday*. The second occurrence of a *wednesday* will become non-supported, but in fact it is reasonable to expect that all the subsequent wednesdays be pointed as non-supported through propagation. This corresponds to considering a detected non-supported action as an automatically deleted action. In this case, it is enough to

reapply the algorithm above after every automatic deletion to detect, recursively, all propagated non-supported actions. The deletion of a detected non-supported action is, however, a "corrective action" to the detection of an time paradox; we have already decided to leave to a user to decide when and how to apply corrective actions, and the process of detecting non-supported actions, even propagated ones, should not change the state of the database. A solution for that would be to fake the deletion of the detected non-supported action during the propagation, *i.e.* consider the detection and propagation of loss of support as a database process—a transaction—that temporarily deletes from the database the detected non-supported action and call Algorithm 5.5 until it returns an empty set, at which point we have reached the end of the propagating process and the temporarily deleted information may be restored. This process may be enriched with the temporary insertion of retroactive actions eventually detected.

To finalise the detection of time paradoxes, since we have already stated that we are not interested in detecting faked executions, all we have to do is show how to detect rule violation. The process is very simple if we restrict ourselves to the detection of violation of executed external actions, for then we may use the information stored in table *executed_action*.

Algorithm 5.7 Detection of rule violation for external actions

```
Input:  update set θ_.
Output:  A set of time-labelled actions
DETECT_VIOLATION(θ_)
BEGIN
VIOLATE := ∅
   For all t : atom in θ_ do
      If PRED(atom) is an external action and
      atom occurs in executed_action(Action, Rule_id, Parms, Time, d+, d_)
         with Time = t
         and Action = atom
         and COND(Rule_id) is satisfied by Parms
      then VIOLATE := VIOLATE ∪{t : atom};
return(VIOLATE);
END
```

If we assume that the table *executed_action* contains all executed external actions, the algorithm clearly detects all rule violations caused by the deletion of an

executed external action. In practice, it is reasonable to expect neither the database nor the auxiliary tables to retain all the information about the past, but just that a "recent" past be kept, while the "distant" past may be periodicaly transferred to tapes and only brought back to the database for special applications; the definition of what "recent" means is clearly a database design decision. As long as the auxiliary tables used here cover the same period of time towards the past as the database itself, the algorithms presented in this section will remain correct; this periodical removal of information from the database into tapes is considered as a huge update that will cause the table *executed_action* to be maintained appropriately, but we may wish to disable the detection of non-supported actions at that time.

All the main algorithms described are to be executed after the database has been updated. They can be executed immediately after the update or after a transaction has committed, in which case the detection of non-supported actions, connected retroactive actions and rule violation can be considered as an independent transaction. This has the advantage that, exept for the computation and storage of temporal dependencies, no extra overhead is placed on the original transaction due to the detection of time paradoxes.

With respect to the worst case complexity of the algorithm for the detection of non-supported actions, we note that if every rule is fired at all times and with temporal dependencies equal to the whole set of time points, the algorithm ends up rechecking every rule at every time point after every update, and therefore degenerating into the naive method for the detection of loss of support. That extreme case, however, appears very unlikely to occur in practice, in which case the described algorithm should perform well. A more detailed analysis of the complexity of the presented algorithms should take in consideration the complexity of the accesses to the database itself, and therefore remains outside the scope of this thesis. The number of rules and the average number of actions executed at each transaction-time should also play a role in determining this complexity. A good evaluator of the efficiency if the detection of loss of support would be the ratio r between the number of non-supported actions detected and the number of rules rechecked; clearly, $0 \leq r \leq 1$, and the closer r is to 1, the smaller is the number of useless rechecks the algorithm performed; to evaluate r, however, it would require having the algorithm implemented in a database system running a real application, which is also outside the scope of this presentation.

This finishes our presentation of the detection of update generated time paradoxes arising from conflicts between the semantics of rule execution and the static

valid-time interpretation of rules in temporal active databases.

Chapter 6

Conclusions

In this chapter the results obtained in this thesis are discussed, compared with the literature and further work based on the results is proposed.

This thesis investigated themes belonging to several areas of Computing Science, that will be revisited in this chapter. The thesis investigates themes in logic and temporal logic, temporal databases and active databases. Moreover, we can see applications of the results obtained here in artificial intelligence and computational linguistics.

All those areas will be separately discussed in this chapter. Our contribution to each one is summarised. Simplifying assumptions in our work are pointed out. Possible extensions and further works will be suggested and related to works published in the literature.

6.1 Overall Analysis of Achievements

The starting point of this work was the analysis of the computational aspects of historical revisionism[1] in temporal active databases. The basic motivation was found in the following question, which is presented here in its generalised form:

Question 6.1 *How is temporal information affected when history is changed?*

The fact that the question was analysed on the sole basis of its applications to the database framework did not imply that we were confined to the realm of databases. In fact, the analysis of this question sent us to explore and extend the formal logic

[1]It was pointed out to me by David Evans that the expression *historical revisionism* is the one actually applied in philosophical and political references to attempts to change the past, or the accepted or official history.

basis of temporal databases, so several contributions of this thesis deal with logic and temporal logic. The double temporality contained in the expression "changing history" led us to study combinations of linear temporal logics with increasing degree of expressivity. Several families of two-dimensional temporal logics were then generated and analysed.

We were then able to come back to the database framework with a formal, two-dimensional temporal logic basis to describe historical updates. This allowed for a characterisation of the differences between the possible semantical interpretations of temporal data.

The task was then to analyse the effects of those updates when the temporal database was extended with temporal rules, generating an interdependency between the data stored in the temporal database. The effects of historical updates were analysed as the time paradoxes arising from the invalidation of the valid-time interpretation under the execution semantics for temporal rules. At this point we were finally able to answer the original motivating question in the temporal active database framework, providing algorithms for the detection of the occurrence of the effects of changing history.

A possible criticism of our approach is that the generality of the presentation of combination of logics in Chapters 2 and 3 does not seem to be justified by our presentation of temporal databases in Chapters 4 and 5. In the pure logical part of this thesis, the results were obtained for temporal logics referring to any class of linear flows of time, but the temporal database models considered later dealt basically with discrete, integer-like flows of time, in both valid-time and transaction-time dimensions; the relevant two-dimensional temporal logics used for the database study were $\bar{\mathsf{U}}\bar{\mathsf{S}}/\mathbb{Z}\times\mathsf{US}/\mathbb{Z}$, $\bar{\mathsf{N}}\bar{\mathsf{P}}/\mathbb{Z}\times\mathsf{US}/\mathbb{Z}$ and, as an intermediary case, $\bar{\mathsf{U}}\bar{\mathsf{S}}/\mathbb{Z}(\mathsf{US}/\mathbb{Z})$ and $\bar{\mathsf{U}}\bar{\mathsf{S}}/\mathbb{Z}\otimes\mathsf{US}/\mathbb{Z}$. The expressivity of both fully two-dimensional systems proved to be adequate for the purposes of our study so we were never forced to chose one, although the logical properties of $\bar{\mathsf{N}}\bar{\mathsf{P}}/\mathbb{Z}\times\mathsf{US}/\mathbb{Z}$ were proved to be nicer.

One possible explanation for that difference in generality is that it is natural to aim at higher generality in the abstract pure logic presentation than that we get in the more application-oriented presentation of databases. While this explanation seems a reasonable one, we argue that there are other reasons supporting the generality of the presentation of combinations of logics. Those reasons are found either in possible extensions of the database concept or in research that has been carried outside the scope of this work.

- In the temporal database presentation the restriction to $\mathbb{Z}$-like valid-time flows was due to a predominance of systems implementing that restriction, and not for any theoretical reasons; this predominance of $\mathbb{Z}$-based systems reflects computational reasons such as efficiency and that real systems clocks are discrete. It is reasonable to expect that if temporal databases become normal, the need for representing dense or real flows of time will be felt, and our logical presentation covers these cases. For example, the axiomatic characterisation of transaction-time databases can be directly applied to those cases, relying only on the linearity of the flow.

- The transaction-time flow of time was chosen to be $\mathbb{Z}$-like due to the discrete nature of update occurrences; however, if we extend the presentation to include the possibility of concurrent transactions, it may become necessary to treat transaction time as dense. Real-time consideration in the execution of transactions may also become an issue, in which case both transaction and valid-time flows may be treated as real.

- Besides considerations of possible extensions of the temporal database, there are reasons for the generality of the presentation of the combination of logics that find their justification outside the scope of this work. In the literature, temporal logic has traditionally dealt with a variety of classes of flows of time and our treatment is a contribution to that tradition.

- Combinations of logics can be seen as a research topic on its own. Other works that deal with the independent combination of monomodal logics [Kracht and Wolter 1991; Fine and Schurz 1991] have also been developed in a framework of the same or even higher generality.

- Also, the idea of the temporalisation process has recently been applied in Computational Linguistics for a different, non-temporal external logic (see discussion below in Section 6.5.2), therefore showing the applicability of our presentation even in areas we had not originally anticipated.

We proceed to analyse the contributions of the thesis to several areas of Computer Science, indicating how our work can be extended and what further work it motivates.

6.2 Contributions to Logic and Temporal Logic

The main focus of this thesis in the area of logic dealt with the combination of two logic systems in order to obtain a new logic system. The issues were:

- Several methods of combination of two logic systems were presented. Each combination involved at least one temporal logic system. Each method had a particular discipline for combining the language, the semantics and the inference system of two logic systems. Each combination generated a single logic system.

- The study of transference of logical properties from the component systems into their combined form has been the major point in the analysis of combination methods. The basic logical properties whose transference was analysed were soundness, completeness and decidability; for some combination methods, the transference of other properties was also investigated such as: the separation property, conservativeness and the compactness property (in the form of strong completeness).

- The investigation of four basic methods has been accomplished. The temporalisation method and the independent combination method were shown to transfer all basic properties, although they do not generate an expressive enough system to be called fully two-dimensional. The full interlacing method does generate a fully two-dimensional temporal system, but in many cases it failed to transfer even the completeness property. As a compromise, it was shown that a restricted interlacing method, although generating two-dimensional temporal logic systems that were not as expressive and generic as the fully interlaced one, accomplishes the transference of all basic logical properties.

Another contribution of our analysis was to answer a question raised by Venema [1990] on the existence of a fragment of the two-dimensional plane temporal logic that, in his own words, was 'better behaved' than the two-dimensional plane system with respect to completeness and decidability properties. We have shown that the two-dimensional temporal logic systems obtained by restricted interlacing are an example of such fragments.

Another question raised by Venema in that same work remains open, namely, whether it is possible to have a complete axiomatisation over the two-dimensional

model using only canonical inference rules, *i.e.* without using the special inference rules **IR1** and **IR2**. This problem seems to be a very hard one. Nevertheless we succeeded in extending Venema's completeness result, that originally holds for only two-dimensional flows built from two identical one-dimensional flows, to any two-dimensional flow built from any flow in the classes $\mathcal{K}_{lin}$, $\mathcal{K}_{dis}$, K_{dense} and $\mathbb{Q}$.

6.2.1 Comparisons, Extensions and Further Work

With respect to combination of logics, the works in the literature that most closely approximate ours in spirit and aims, are those of Kracht and Wolter [1991] and of Fine and Schurz [1991]. Both works concentrated on monomodal logics, and investigated the transference of logical properties for only the method we called here independent combination. However, their work investigated several paths that suggest that further work may be done in our studies. First, they analysed the transference of many other properties from two logic systems to its combined form, *e.g.* finite model property and interpolation. Second, both works did not concentrate only in linear systems and they were able to extend their results to any class of underlying Kripke frames. Third, Fine and Schurz's work generalised the independent combination method to more than two monomodal logics.

Those two papers cited above therefore suggest several extensions to our work. Note, however, that the temporalisation method was easily shown to be extensible to many temporal logic systems in Example 2.4. The focus on linear flows of time was due to the later application to linear database systems, but we believe that this restriction may be lifted without damaging the transference results of the temporalisation and independent combination methods. These have to be further investigated and the transference of any other logical property has to be analysed on its own.

The generalisation of combination methods other than the independent combination method to modal logics is another area for further work. As noted in Chapter 2, the temporalisation process is directly extensible to monomodal logics. It may even be the case that, for monomodal logics, the full interlacing method achieves transference of completeness over several classes of fully two-dimensional Kripke frames using only canonical inference rules, as it is suggested by the results in [Segerberg 1973].

Note that all the systems dealt with in Chapters 2 and 3 were extensions of classical logic. It is possible that the temporalisation process preserves its transference properties even in the case the underlying system is not an extension of classical

logic. What if the external temporal logic is non-classical itself? The same question applies to other combination methods. Do they achieve transference of logical properties when one or both of the combined temporal of modal logics is not classical? Gabbay [1992] has recently posed that question in a very generic framework involving Labelled Deductive Systems (LDS) and found that in order to obtain the transference of completeness we do not need the full power of classical logic but only some weaker form of monotonicity. He has also developed other methods of combination called *fibring* that depends on the choice of a fibring function. A fibring function maps the truth value of atoms in one logic's semantics with the semantics of formulae in other logic's semantics. Gabbay's *dovetailing* process, obtained with a certain class of fibring functions, is similar to the independent combination method extended to logics respecting those weaker conditions of monotonicity. More work on this area is needed to clarify exactly how fibring is related to existing combination methods.

There are also other possible types of combinations of one-dimensional temporal logics that may be explored. As pointed out in the end of Chapter 4, two linear flows of time can be merged into another one; the question is then how to combine two one-dimensional temporal logics into another one-dimensional temporal logic over the merged flow.

6.3 Contributions to Temporal Databases

Besides the work on logic and temporal logic, the other topic of this thesis was database theory, focusing on the dynamic aspects of temporal databases (Chapter 4) and on temporal active databases (Chapter 5).

Temporal databases were formally presented over a first-order temporal logic framework; safe temporal formulae were adopted as queries, which allowed a propositional abstraction over the first-order presentation, enabling us to apply the propositional two-dimensional framework we had developed to the study of temporal databases. The main achievements then dealt with the dynamics of temporal databases in the following ways.

- The two-dimensional plane model was used to describe the semantics of updates in temporal databases.
- A formal characterisation of the difference between the transaction-time and

valid-time databases was obtained by means of an abstract, axiom-based description over the two-dimensional framework. This description did not rely on the details of how updates were actually done, but rather stressed only the effects of updates.

6.3.1 Further Work

The two-dimensional plane temporal model and the several logics associated to it can be seen as a formal basis for temporal databases that store both valid and transaction times. This kind of *two-dimensional temporal database* (or *bitemporal database*, as called in [Jensen *et al.* 1992]) was described by Snodgrass and Ahn [1985], where every piece of data is associated to two time stamps, one representing the user controlled period of history to which the data refers to, the other representing the period between the insertion and deletion of the data. In this case, a two-dimensional algebra is needed as a counterpart of the two-dimensional temporal logic based calculus.

In a survey of temporal algebras, McKenzie and Snodgrass [1991] reported basically only one-dimensional temporal algebras. Our work with the composition of temporal logics suggests that a family of two-dimensional temporal algebras might be obtainable by an analogous composition of existing one-dimensional ones. One interesting property to be studied then would be the equivalence of query expressivity between the algebra and the logic, in the same lines that query expressivity was shown to be equivalent for the non-temporal relational calculus and algebra.

The temporal data representation and the temporal query language of our exposition can be extended. The temporal database was restricted to contain only temporally bounded discrete information represented by labelled atoms, where the labels were conjunctions of terms of the form $t = t_0$, $t < t_0$ and $t_0 < t$, for t a term variable and t_0 a term constant. Kabanza, Stevenne and Wolper [1990] have proposed a more expressive data representation called *linear repetitive points*, or l.r.p.'s, and showed them to have the same expressivity as formulae labelled with Pressburger arithmetic expressions, *i.e.* first-order formulae over the signature containing predicate symbols $=$ and $<$ and function symbols *suc* and $+$ [Boolos and Jeffrey 1989]. Furthermore, the associated query language was not based on temporal logic, but consisted of a two-sorted first-order language, one-sort for time and the other sort for domain elements.

Even with our less expressive data representation, it is possible to extend the expressiveness of the temporal query languages (the contrasts between temporal data

expressivity and temporal query expressivity were studied in [Baudinet, Niezette and Wolper 1991]). Chomicki and Imieliński [1988] use a two-sorted datalog-style language for temporal recursive queries, adding deductive rules to the database that are declaratively interpreted. An equivalently expressive deductive query language involving temporal operators and recursion can be found in [Abadi and Manna 1989].

The two-dimensional description of database updates is independent of the data representation. To extend our results from temporally bounded data to l.r.p.'s we need only find an equivalent version of Proposition 4.2 dealing with l.r.p.'s and l.r.p. updates. The expressivity of the query language does not affect the dynamics of temporal databases either, so the two-dimensional model of database evolution could go along more expressive data representation and query languages without major changes. In fact, the two-dimensional model is based solely on the linearity of the valid-time flow of time and, although the majority of temporal database systems surveyed in [McKenzie and Snodgrass 1991] deals with discrete, integer-like flows of time, the model and even the axiomatic distinction between transaction-time and valid-time databases must hold over dense and continuous valid-time flows of time (provided the transaction flow of time remains integer-like).

A totally unmentioned but very important theme in the study of temporal databases is their physical implementation. File organisation and special indexing strategies for temporal data are important issues to take into consideration for efficient querying and updating of temporal databases. Unfortunately, these subjects lie outside the scope of this work; refer to [Gunadhi and Segev 1993; Tansel *et al.* 1993] for an account of some recent developments in this area.

To finalise, we would like to comment on a recently published book containing a collection of papers on temporal databases [Tansel *et al.* 1993], which came to our knowledge by the time of finishing this work. The book is divided in four parts. In the first part, several temporal extensions of the relational data model are presented; several distinct temporal query languages, temporal algebras and data representation are presented; and different temporal ontologies are considered, such as point based and interval based flows of time. The second part is concerned with temporal extensions of non-relational data models, *e.g.* the object-oriented, the extended entity-relationship, time sequences and the deductive data models. The third part deals with implementation issues such as query processing, optimisation, indexing and storage strategies. The last part deals with other temporal database related issues, such as temporal knowledge bases for simulations, heterogeneous environmentsand temporal reasoning in general. Particularly conspicuous is the

absence in that book of any paper dealing with the problem of updating temporal databases, which is our main concern in this work. The book is, however, a good reference for much of the data modelling and implementation work that has been done in the field of temporal databases.

6.4 Contributions to Temporal Active Databases

The framework of temporal databases was shifted from a passive repository to an active one with the addition of temporal rules, which followed the imperative future paradigm. This work contributions to the area of temporal active databases were as follows:

- A two-dimensional view of the imperative future paradigm allowed for the generalisation of the format of temporal rules, eliminating the restriction "past and present implies future"; the effects of that restriction had already been discussed in [Manning and Torsun 1989].

- Two distinct semantics for temporal rules, one imperative and the other declarative, were contrasted and as a result several "time paradoxes" were classified. The imperative semantics unfolded over the two-dimensional model, while the valid-time one is one-dimensional. The classification of time paradoxes was based on the two-dimensional model.

- Algorithms for the detection of some of the occurrence of time paradoxes were proposed. It was not require that the second (transaction-time) dimension be fully stored in the database. Only transaction-time information related to the firing and execution of rules was needed.

6.4.1 Further Work

Several implementation issues of temporal active databases remain to be investigated. The most expensive of all involved operations is the evaluation of the condition part of rules. This can be done more efficiently than the naive evaluation of all condition parts at all transaction times by avoiding recomputation of unchanged queries. For non-temporal rules, *Rete* [Forgy 1982] is the most usual algorithm coming from artificial intelligence applications of production systems and expert systems. When transposed to relational databases, the Rete algorithm was shown

to have several deficiencies, so other algorithms were proposed aiming at the reduction of recomputation [Sellis, Lin and Raschid 1993]. The Rete algorithm was extended to finite hierarchical possible worlds [Cavalcanti 1993], but temporal applications so far seem to rely basically on the naive evaluation of the condition part of all rules at every transaction time [Loucopoulos *et al.* 1990; Barringer *et al.* 1989; Manning and Torsun 1989].

A natural extension of temporal rules as previously presented, which actually has an impact on the efficiency of the execution of rules, is the addition of triggers to temporal rules. Triggers are (possibly 0-ary) predicates that are placed as a label of the rule

$$trigger : Cond \Rightarrow Action,$$

so that *Cond* will be evaluated only when *trigger* is active. Triggers are normaly represented in rule based languages in the format

when *Trigger* if *Condition* then *Action*

as, for example, in the ERL-language of [McBrien *et al.* 1991]. Triggers remain active for only one evaluation cycle which, in terms of the two-dimensional model, corresponds to the persistence of the triggers for only one unit over the transaction flow of time, ie they may hold only at the diagonal points; except for that peculiarity, triggers can be seen as a conjunt of the condition part of a rule. Their effect, however, is felt in the efficiency of rule based systems, for in each evaluation cycle, only the triggered rules will have their *Condition*-part evaluated, saving a lot of time. In the TEMPORA system [Loucopoulos *et al.* 1990], rule triggers are called *flows* and can be either external, *i.e.* activated by an external user agent, or internal, *i.e.* activated by the action part of a rule, or they can be event driven triggers, *i.e.* activated by the ticking of the system's clock or the updating of a relation. Flows can also be combined with boolean operators to form more complex triggers.

The notion of a *transaction* is one that deserves a special treatment in active valid-time databases, but remained outside the scope of our presentation. The traditional view of a transaction is one of a "logical unit of work" in the database, that must either be entirely completed (when the transaction commits) or all its effects must be removed from the database (when the transaction is rolled back). The problem with active valid-time databases is that external actions executed in the past cannot be rolled back. Therefore, if a transaction takes more than one unit of time to be finished, the "logical unit of work" may be violated due to the

impossibility of rolling it back. The solution in the TEMPORA system was to limit transactions to a single tick. With this constraint, a transaction is initiated by activating a special kind of trigger, called external flow; the transaction goes on by repeating the execution cycle in the same valid time, until there are no actions to execute, in which case the transaction commits and all the external actions that were fired are sent to the database environment; if, during the execution cycles, the transaction is aborted, no external action has been sent to the environment, so the transaction may be rolled back.

Further work is being developed in the design, capturing and structuring of rules that remains outside the scope of this work; such work is very important in making the rule-based approach easier to manipulate from a programmer's point of view. On the system's side, a topic that deserves further consideration is how to overcome the constraint of transactions lasting at most one time unit, so that the notion of a rule-based *temporal transaction* may be formed.

6.5 Other Contributions

6.5.1 Artificial Intelligence

Belief revision is a topic of study of artificial intelligence and cognitive sciences that deals with the problem of changing one's idea about the world. Most approaches to this problem have focused on the non-monotonic aspects of this change of belief, *i.e.* the set of inferred data is not always preserved when a new piece of data is included to the original premises. Since we are working with "changes in history", our work has some features that contribute to the study of belief revision:

- With the temporalisation process, we have contributed in explicating the intrinsic temporality in the change of belief.
- The two-dimensional temporal evolution provides a framework which allows for changes in history without giving up monotonicity, in the sense that the extra dimension preserves the set of all conclusions reachable at each transaction-time. The persistence of unchanged data from one transaction time to the next one is obviously non-monotonic.

The two-dimensional model can also serve as part of a data model for expert systems that are supposed to give explanations of past actions. In this case, the systems ability to retrieve not only the current state of history, but also the history

as seen at previous moments, can provide enough basis to explain why things were *then* done in a way for which *now* there seems to be no rational support.

Furthermore, the rule based approach arround which we build the temporal links, and for which we developed the detection of changes in the past may be adapted to expert systems. For instance, if an expert system provides advice on some domain of knowledge with temporal data (*e.g.* an investment advisor), changes in the temporal data (*e.g.* correction of data, discovery of frauds, release of new information about the past) may lead to non-supported actions (*e.g.* the retraction of earlier advice) which can then propagate to the present (*e.g.* the generation of new advice). The inclusion of such a capability in existing systems, such as PAYE or METATEM, is subject to further investigation.

Another possible line of research is to extend the two-dimensional system to several temporal agents. As it is, the system copes only with the evolution of one agent's beliefs (*i.e.* the view the database manager has of an evolving history). We could then think of several agents with communicating capabilities and with internal 'extra dimensions' capable of recording the evolution of beliefs of the other agents. This would imply a combination of techniques of distributed AI, reasoning with incomplete information and the two-dimensional approach, which seems no trivial task; therefore, we do not suggest here that such an integration may be done in a straightforward way, but rather that this is a subject for a substantial amount of research.

6.5.2 Computational Linguistics

Our work is connected to computational linguistics in a few ways. The very idea of two-dimensional temporal logics first appeared in the literature in the analysis by Kamp [1971] of the temporal meaning of the word "now" in gramatical sentences.

Recently, the ideas of the temporalisation process have been adopted by computational linguists as a form of *layering*. Blackburn, Gardent and Meyer-Viol [1993] follow the same principles of the temporalisation process in order to provide the right level of expressive power needed to model many grammar formalisms. They first define L^{T}, a modal logic to describe constraints on grammar parsing trees; they then describe how to decorate the parsing tree with feature structures described by a generic *feature logic*, L^{F}, generating the combined logic $\mathsf{L}^{\mathsf{T}}(\mathsf{L}^{\mathsf{F}})$ by means of combining their syntax, semantics and inference systems. They show how such combined logic $\mathsf{L}^{\mathsf{T}}(\mathsf{L}^{\mathsf{F}})$ can be used to model an existing linguistic theory.

In the same lines of combination of logics, but perhaps going even further in the

interaction between the two component logics, we found the brand new ideas of *fibred semantics* being applied as a combination of computational linguistic frameworks. In a still unpublished paper [Dörre, Gabbay and König 1993], the ideas of fibring the Lambek Calculus with feature logics are developed, and an example case is provided with the combination of the Lambek Calculus and Horn-clause logic programming.

We can only wait to see what other new applications of the ideas in this work will appear in the future, among the computational linguistics community or elsewhere.

Appendix A

Algorithms

This appendix presents all the algorithms developed in Chapter 5.

A.1 Auxiliary algorithms

We present here the alogorithms that manipulate the data structures

$syntactical\_dependences(Pred\_Name, Pos\_dep\_list, Neg\_dep\_list)$
$executed\_action(Action, Rule\_id, Parms, Time, d_+, d_-)$

The first two algorithms manipulate the first data structure fo the retrieval of the rules affected by a given positive/negative syntactical dependence.

Algorithm A.1 Positive syntactical dependences (equivalent to Algorithm 5.1)

```
Input:  a predicate name
Output:  a list of rule identifiers
   in which the predicate appear as
   a positive syntactical dependence

POSSYNT(PredName)
BEGIN
Select Pos_dep_list from syntactical_dependences
   where Pred_Name = PredName;
return Pos_dep_list;
END
```

Algorithm A.2 Negative syntactical dependences (equivalent to Algorithm 5.2)

```
Input:  a predicate name
Output:  a list of rule identifiers
   in which the predicate appear as
   a negative syntactical dependence

NEGSYNT(PredName)
BEGIN
Select Neg_dep_list from syntactical_dependences
   where Pred_Name = PredName;
return Neg_dep_list;
END
```

The following algorithms manipulate the second data structure for the retrival of executed actions that may have been affected by an update.

Algorithm A.3 Positive intersection rows (equivalent to Algorithm 5.3)

```
Input:  a rule ID and a list of time points.
Output:  a set of rows from executed_action table
   with the same rule ID, RID,
   and positive dependence overlapping TIMES.
POSROWS(RID, TIMES)
BEGIN
ROWS := ∅;
For every row of the table given by
   Select Action, Rule_id, Parms, Time, d+, d-
   from executed_action
   where RID = Rule_id and d+∩ TIMES ≠ ∅
   do    ROWS := ROWS ∪{row(Action, Rule_id, Parms, Time, d+, d-)}
return ROWS;
END
```

Algorithm A.4 Negative intersection rows (equivalent to Algorithm 5.4)

```
Input:  a rule ID and a list of time points.
Output:  a set of rows from executed_action table
```

```
   with the same rule ID, RID,
   and negative dependence overlapping TIMES.

NEGROWS(RID, TIMES)
BEGIN
ROWS := ∅;
For every row of the table given by
   Select Action, Rule_id, Parms, Time, d+, d-
   from executed_action
   where RID = Rule_id and d- ∩ TIMES ≠ ∅
   do    ROWS := ROWS ∪{row(Action, Rule_id, Parms, Time, d+, d-)}
return ROWS;
END
```

A.2 Main Algorithms

We present here a combination of the main algorithm to detect non-supported actions, Algorithm 5.5, and its extension to detect connected retroactive actions, Algorithm 5.6. Note that the output now is a pair of sets, namely the set of detected non-supported actions and the set of connected retroactive actions, generated by a recent update.

Algorithm A.5 Detection of non-supported and connected retroactive actions

```
Input:  update sets θ+ and θ-.
Output:  A pair containing a set of time-labelled non-supported actions
   and a set of retroactive actions.

DETECT_NONSUP(θ+, θ-)
BEGIN
NONSUP := ∅;
RETRO := ∅;

/* The first part of the algorithm deals with insertions */
For every times : atom in θ+
BEGIN
   For every rule id RID in NEGSYNT( PREDNAME( atom ) )
```

```
   BEGIN
      For each row(Action, RID, Parms, Time, d+, d-)
      in NEGROWS( RID, times )
      BEGIN
         If the query COND( RID ) is not satisfied by Parms at Time
         and Time : Action holds in the database
         then
         BEGIN
            NONSUP := NONSUP ∪{Time : Action};
            delete the row from the executed_action table.
            /* OBS: The detection of retroactive actions */
            /* is inserted here (Algorithm 5.6)*/
            If the query COND( RID ) is satisfied at Time for Parms' ≠ Parms
               and there is no row in table executed_action such that
               it contains Action(Parms'), ttRID, Parms' and Time
            then
               RETRO := RETRO ∪{Time : Action(Parms')};
         END
         else if COND( RID ) is satisfied with new temporal
            dependences d'+ ≠ d+ or d'- ≠ d- ,
            then modify the row in the executed_action table
            with dependences d'+ and d'- .
      END
   END
END


/* The second part of the algorithm deals with deletions */
For every times : atom in θ-
BEGIN
   For every rule id RID in POSSYNT( PREDNAME( atom ) )
   BEGIN
      For each row(Action, ttRID, Parms, Time, d+, d-)
      in POSROWS( RID, times )
      BEGIN
         If the query COND( RID ) is not satisfied by Parms at Time
```

```
            and Time : Action holds in the database
            then
            BEGIN
               NONSUP := NONSUP ∪{Time : Action};
               delete the row from the executed_action table.
               /* OBS: The detection of retroactive actions */
               /* is inserted again here (Algorithm 5.6) */
               If the query COND( RID ) is satisfied at Time for Parms' ≠ Parms
                  and there is no row in table executed_action such that
                  it contains Action(Parms'), ttRID, Parms' and Time
               then
                  RETRO := RETRO ∪{Time : Action(Parms')};
            END
            else if COND( RID ) is satisfied with new temporal
               dependences d'+ ≠ d+ or d'- ≠ d- ,
               then modify the entry in the table
               with dependences d'+ and d'- .
         END
      END
   END
END
return( (NONSUP, RETRO) )
END
```

The final algorithm deals with the detection of rule violations.

Algorithm A.6 (equivalent to 5.7) Detection of rule violation for external actions

```
Input:  update set θ- .
Output:  A set of time-labelled actions

DETECT_VIOLATION(θ-)
BEGIN
VIOLATE := ∅;
   For all t : atom in θ- do
      If PRED(atom) is an external action and
      atom occurs in executed_action(Action, Rule_id, Parms, Time, d+, d-)
         with Time = t
         and Action = atom
```

```
          and COND(Rule_id) is satisfied Parms
       then VIOLATE := VIOLATE ∪{t : atom};
return(VIOLATE);
END
```

Appendix B

Auxiliary Proofs

This appendix presents proofs of some auxiliary or secondary lemmas and propositions cited in the body of the thesis.

Two-dimensional completeness

Theorem B.1 (2D-completeness) (Theorem 3.3)

There are sound and complete axiomatisations over the two-dimensional plane classes $\mathcal{K}_{dis}\times\mathcal{K}_{dis}$, $\mathbb{Q}\times\mathbb{Q}$, $\mathcal{K}_{lin}\times\mathcal{K}_{dis}$, $\mathcal{K}_{lin}\times\mathbb{Q}$ and $\mathcal{K}_{dis}\times\mathbb{Q}$.

Proof We prove completeness over $\mathcal{K}_{dis}\times\mathbb{Q}$, the other cases being simplifications of this one. For that, on the horizontal dimension we add the discreteness axiom

Dis-a $F\top\rightarrow U(\top,\bot)$

together with its mirror image **Dis-b**. On the vertical dimension we add axioms for denseness and no end points:

Den-a $\neg\overline{U}(\top,\bot)$
Noe-a $\overline{F}\top$

together with their mirror images **Den-b** and **Noe-b**. The construction of IR*-theories has to take into account the extra axioms, but except for that it is completely analogous to that over $\mathcal{K}_{lin}\times\mathcal{K}_{lin}$ in Section 3.3.1.

Let Γ_0 be a set of formulae consistent with the two-dimensional axiomatisation over $\mathcal{K}_{dis}\times\mathbb{Q}$. Since this axiomatisation extends that over $\mathcal{K}_{lin}\times\mathcal{K}_{lin}$, the gap and corner filling lemmas still hold and the construction of the grid is totally analogous of the previous one. So we obtain a two dimensional grid $\mathcal{G}^*=(X^*,<^*,\overline{X}^*,\overline{<}^*,f^*)$ such that there are no counterexamples left in $\mathcal{G}^*$ and $(X^*,<^*)$,$(\overline{X}^*,\overline{<}^*)\in\mathcal{K}_{lin}$

and $\mathcal{G}^*$ satisfies **C1, C2a, C2b, C3a, C3b, C4a, C4b, C5a, C5b** as defined in Section 3.3.1.

It remains only to be proved that $(X^*, <^*) \in \mathcal{K}_{dis} \subseteq \mathcal{K}_{lin}$ and $(\overline{X}^*, \overline{<}^*)$ is isomorphic to $\mathbb{Q}$. For that, let $x \in \overline{X}^*$ and $t \in X^*$. Suppose t is not the last element of X^*, so that, by **C2a**, $F\top \in f^*(t,x)$; by axiom **Dis-a**, $U(\top,\bot) \in f^*(t,x)$. Then, by **C4a**, it follows that there exists $s \in X^*$, $t <^* s$, such that $\top \in f^*(s,x)$ and for all u, $t <^* u <^* s$, $\bot \in f^*(u,x)$, *i.e.* there exists no such $u \in X^*$ between t and s. So s is the successor of t. Analogously, **Dis-b** and **C4b** gives us a predecessor of t. We have thus proved that $(X^*, <^*) \in \mathcal{K}_{dis}$.

By Axiom **Den-a**, it follows that for every $x \in \overline{X}^*$ and $t \in X^*$, $\neg\overline{U}(\top,\bot) \in f(t,x)$ and by condition **C4a** one of the following must hold:

- There is no y, $x \overline{<}^* y$ with $\top \in f(t,y)$. This contradicts **Noe-a**; similarly, **Noe-b** yields no initial points. So
- For every y, $x \overline{<}^* y$, there exists z, $x \overline{<}^* z \overline{<}^* y$, $\bot \notin f(t,z)$ (which is always true); in other words, $(\overline{X}^*, \overline{<}^*)$ is dense.

The flow $(\overline{X}^*, \overline{<}^*)$ is therefore linear, dense with no end points and, by construction, it is also countable. So $(\overline{X}^*, \overline{<}^*)$ is isomorphic to $\mathbb{Q}$. This ends the proof. □

Normal form for $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$

Lemma B.1 (Lemma 3.6)
Let A be a formula of $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$. There exists a normal form formula A^* equivalent to A, such that all the occurrences of $\overline{\bigcirc}$ and $\overline{\bullet}$ in it are in the form $\overline{\bigcirc}^k p$ and $\overline{\bullet}^l q$, where p and q are atoms.

Proof First we show that converse of the interlacing axioms are theorem too. For that, note that U and S respect the *congruence property*, *i.e.* if $A \leftrightarrow C$ and $B \leftrightarrow D$ then $U(A,B) \leftrightarrow U(C,D)$ and $S(A,B) \leftrightarrow S(C,D)$. Also note that

equiv $\quad (p \leftrightarrow \overline{\bigcirc}\,\overline{\bullet} p) \wedge (p \leftrightarrow \overline{\bullet}\,\overline{\bigcirc} p)$

The transitivity of $\rightarrow$ connects the steps in the proof of $U(\overline{\bigcirc}p, \overline{\bigcirc}q) \rightarrow \overline{\bigcirc}U(p,q)$ below:

$$
\begin{array}{lll}
U(\overline{\bigcirc}p, \overline{\bigcirc}q) \rightarrow \overline{\bigcirc}\,\overline{\bullet}\,U(\overline{\bigcirc}p, \overline{\bigcirc}q) & & \text{by } \textbf{equiv} \\
\rightarrow \overline{\bigcirc}U(\overline{\bullet}\,\overline{\bigcirc}p, \overline{\bullet}\,\overline{\bigcirc}q) & & \text{by interlacing axiom} \\
\rightarrow \overline{\bigcirc}U(p,q) & & \text{by } \textbf{equiv} \text{ and congruence}
\end{array}
$$

It follows that $U(\overline{\bigcirc}p, \overline{\bigcirc}q) \leftrightarrow \overline{\bigcirc}U(p,q)$. It is completely analogous to show the converse of other interlacing axioms, so we omit the details.

Given A in the language of $\mathsf{US} \times \bar{\mathsf{N}}\bar{\mathsf{P}}$, the equivalence between both sides of the interlacing axioms allows for "pushing in" the vertical operators $\overline{\bigcirc}$ and $\overline{\bullet}$, so a simple induction on the number of nested temporal operators in A shows an algorithmic way to generate an equivalent formula A^* in the desired normal form. □

Equivalence of diagonal axioms

Lemma B.2 (Lemma 4.1)
Consider the formulae

D1	$\Diamond\delta \wedge \overline{\Diamond}\delta$		**d1** $\Diamond\delta$
D2	$\delta \rightarrow (G\neg\delta \wedge H\neg\delta \wedge \overline{G}\neg\delta \wedge \overline{H}\neg\delta)$	*and*	**d2** $\delta \rightarrow (G\neg\delta \wedge H\neg\delta)$
D3	$\delta \rightarrow (\overline{H}G\neg\delta \wedge \overline{G}H\neg\delta)$		**d3** $\delta \leftrightarrow \overline{\bigcirc}\bigcirc\delta$

Let $\mathcal{M}$ be a two-dimensional plane model over $\mathbb{Z}\times\mathbb{Z}$. Then the formula **D1**$\wedge$**D2**$\wedge$**D3** *holds over $\mathcal{M}$ iff* **d1**$\wedge$**d2**$\wedge$**d3** *holds over $\mathcal{M}$.*

Proof By Lemma 3.7 we know that **D1**$\wedge$**D2**$\wedge$**D3** holds over $\mathcal{M}$ iff the relation i defined as below

$$i = \{(t,x) \in \mathbb{Z}\times\mathbb{Z} \mid \mathcal{M}, t, x \models \delta\}.$$

is an isomorphism in $\mathbb{Z}$. So all we have to do is to prove that i as defined above is an isomorphism iff **d1**$\wedge$**d2**$\wedge$**d3** holds over $\mathcal{M}$. The *only if* is a straightforward verification that for all x and t in $\mathbb{Z}$, $\mathcal{M}, t, x \models$ **d1**$\wedge$**d2**$\wedge$**d3**.

Assume **d1**$\wedge$**d2**$\wedge$**d3** holds over $\mathcal{M}$. Then:

1. **d1** gives us that for every x there exists a t such that $\mathcal{M}, t, x \models \delta$;
2. **d2** gives us that for every x, t, t', $t \neq t'$, $\mathcal{M}, t, x \models \delta$ implies $\mathcal{M}, t', x \not\models \delta$;
3. **d3** give us that for every x, t, $\mathcal{M}, t, x \models \delta$ iff $\mathcal{M}, t+1, x+1 \models \delta$ iff for every $n \in \mathbb{Z}$, $\mathcal{M}, t+n, x+n \models \delta$

The first two items give us that $i^{-1} : \mathbb{Z} \rightarrow \mathbb{Z}$ is a function. To show that i is also a function, suppose that $(t, x_1), (t, x_2) \in i$. By linearity of $\mathbb{Z}$, it follows that either $x_1 < x_2$ or $x_2 < x_1$ or $x_1 = x_2$. Let $x_1 - x_2 = \mathrm{m}$; then, by the third item above, $(t+m, x_2 + m = x_1) \in i$, so $t = (t+m)$ and $m = 0$. It follows that $x_1 = x_2$, so $i : \mathbb{Z} \rightarrow \mathbb{Z}$ is a function. Directly by the definition of i, it follows that i is a bijection.

Again by the third item above, if $i(t_1) = x_1$ and $i(t_2) = x_2$, then $t_1 - t_2 = x_1 - x_2$. It follows that i is order preserving and hence an isomorphism, which finishes the proof. □

The meta-level axiom Roll

Lemma B.3 (Lemma 4.2)
Consider the meta-level axiom

Roll $((\delta \vee F\delta) \wedge q) \rightarrow \overline{\bigcirc} q$.

If **Roll** *holds over a two-dimensional $\mathcal{M}$ for any literal q, it also holds over $\mathcal{M}$ for any* US*-formula that does not contain future operators, i.e. does not contain U and its derived operators.*

Proof By induction on the length of A, where A is a formula that does not contain U and its derived operators and the length of a formula is the number of symbols in it. The cases $A = q$ and $A = \neg q$ are given by the fact that **R**oll holds over $\mathcal{M}$.

If $A = \neg\neg B$, then $\mathcal{M}, t, x \models \neg\neg B$ iff $\mathcal{M}, t, x \models B$. By the induction hypothesis, $\mathcal{M}, t, x + 1 \models B$, so $\mathcal{M}, t, x \models ((\delta \vee F\delta) \wedge A) \rightarrow \overline{\bigcirc} A$.

If $A = B \wedge C$, then from $\mathcal{M}, t, x \models B \wedge C$ it follows $\mathcal{M}, t, x \models B$ and $\mathcal{M}, t, x \models C$ and hence, by induction hypothesis, $\mathcal{M}, t, x + 1 \models B$ and $\mathcal{M}, t, x + 1 \models C$, so $\mathcal{M}, t, x \models ((\delta \vee F\delta) \wedge A) \rightarrow \overline{\bigcirc} A$.

If $A = \neg(B \wedge C)$, then $\mathcal{M}, t, x \models \neg B$ and $\mathcal{M}, t, x \models \neg C$ and hence, by induction hypothesis, $\mathcal{M}, t, x+1 \models \neg B$ or $\mathcal{M}, t, x+1 \models \neg C$, so $\mathcal{M}, t, x \models ((\delta \vee F\delta) \wedge A) \rightarrow \overline{\bigcirc} A$.

If $A = S(B, C)$, then from $\mathcal{M}, t, x \models (\delta \vee F\delta) \wedge S(B, C)$ it follows that $t \leq x$ and that there exists $s < t$ such that $\mathcal{M}, s, x \models B$ and for every u, $s < u < t$ $\mathcal{M}, u, x \models C$. Then induction hypothesis, $\mathcal{M}, s, x+1 \models B$ and for every u, $s < u < t$ $\mathcal{M}, u, x + 1 \models C$. Hence $\mathcal{M}, t, x \models ((\delta \vee F\delta) \wedge A) \rightarrow \overline{\bigcirc} A$.

Finally, if $A = \neg S(B, C)$, then we have to consider two cases. One possibility is that $\mathcal{M}, t, x \models (\delta \vee F\delta) \wedge H\neg B$. On the other hand, if there exists $s < t$ such that $\mathcal{M}, s, x \models B$ then there exists s', $s < s' < t$ such that $\mathcal{M}, s', x \models \neg B$ and $\mathcal{M}, s', x \models \neg C$ and for all u, $s' < u < t$, $\mathcal{M}, u, x \models \neg B$. In both cases, the induction hypothesis leads us to $\mathcal{M}, t, x \models ((\delta \vee F\delta) \wedge A) \rightarrow \overline{\bigcirc} A$. □

Bibliography

Abadi, M. and Z. Manna [1989]. "Temporal Logic Programming." *Journal of Symbolic Computation*, 8:277–295.

Abiteboul, S. and V. Vianu [1991]. "Datalog Extensions for Database Queries and Updates." *J. of Computer and System Sciences*, 43:62–124.

Aqvist, L. [1979]. "A Conjectured Axiomatization of Two-dimensional Reichenbachian Tense Logic." *J. of Philosophical Logic*, 8:1–45.

Barringer, H., M. Fisher, D. M. Gabbay, G. Gough, and R. P. Owens [1989]. "METATEM: A Framework for Programming in Temporal Logic." In *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness*, volume 430 of *LNCS*, pages 94–129. Springer-Verlag, Mook, Netherlands.

Barringer, H., M. Fisher, D. Gabbay, and A. Hunter [1991]. "Meta-Reasoning in Executable Temporal Logic." In *Second Conference on the Principles of Knowledge and Reasoning*, pages 40–49, San Mateo, California. Morgan Kaufmann.

Baudinet, M., M. Niezette, and P. Wolper [1991]. "On the Representation of Infinite Temporal Data and Queries." In *1Oth ACM Symposium on Principles of Database Systems*, pages 280–290.

Blackburn, P., C. Gardent, and W. Meyer-Viol [1993]. "Talking About Trees." In *6th Conference of the European Chapter of the Association of Computational Linguistics*, pages 21–29.

Boolos, G. S. and R. C. Jeffrey [1989]. *Computability and Logic*. Cambridge University Press, third edition.

Burgess, J. P. and Y. Gurevich [1985]. "The Decision Problem for Linear Logic." *Notre Dame Journal of Formal Logic*, 26(2):566–582.

Burgess, J. P. [1982]. "Axioms for Tense Logic I: "Since" and "Until"." *Notre Dame Journal of Formal Logic*, 23(4):367–374.

Burgess, J. P. [1984]. "Basic Tense Logic." In Gabbay, D. and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 89–133. D. Reidel Publishing Company.

Casanova, M. A. and A. L. Furtado [1982]. "A Family of Temporal Languages for the Description of Transition Constraints." In *3rd Worksohp on Logical Bases for Databases*, Toulouse, France.

Cavalcanti, M. [1993]. "$\mathcal{PW}$-XRete: The Possible World in Real Life." In *Proc. International Joint Conference of Artificial Intelligence.*

Chomicki, J. and T. Imieliński [1988]. "Temporal Deductive Databases and Infinite Objects." In *7th ACM Symposium on Principles of Database Systems*, pages 61–73, Austin, Texas.

Chomicki, J. and D. Niwiński [1993]. "On the Feasibility of Checking Temporal Integrity Constraints." In *12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.*

Codd, E. F. [1970]. "A Relational Model for Large Shared Data Banks." *Communications of the ACM*, 13(6):377–387.

Codd, E. F. [1972]. "Relational Completeness of Data Base Sublanguages." In Rustin, R., editor, *Data Base Systems*, pages 65–98. Prentice-Hall, Englewood Cliffs, New Jersey.

Dörre, J., D. Gabbay, and R. König. "Fibred Semantics for Feature-Based Grammar Logic." Preliminary unpublished draft — 28 Jully, 1993, 1993.

Fine, K. and G. Schurz. "Transfer theorems for stratified multimodal logics." Unpublished manuscript, 1991.

Finger, M. and D. M. Gabbay [1992a]. "Adding a Temporal Dimension to a Logic System." *Journal of Logic Language and Information*, 1:203–233.

Finger, M. and D. M. Gabbay [1992b]. "Updating Atomic Informations in Labelled Database Systems." In *4th International Conference on Database Theory*, pages 188–200, Berlin.

Finger, M., M. Fisher, and R. Owens [1993]. "METATEM at work: Modelling Reactive SystemsUsing Executable Temporal Logic." In *Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh.

Finger, M., P. McBrien, and R. Owens [1991]. "Databases and Executable Temporal Logic." In *Annual Esprit Conference*, pages 288–302. Comission of the European Communities.

Finger, M. [1992]. "Handling Database Updates in Two-dimensional Temporal Logic." *J. of Applied Non-Classical Logic*, 2(2):201–224.

Forgy, C. L. [1982]. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem." *Artificial Intelligence*, 19.

Frege, G. [1879]. "Begriffsschrift." In [Heijennort 1982], pages 1–82. Harvard University Press.

Gabbay, D. M. and I. M. Hodkinson [1990]. "An axiomatization of the temporal logic with Until and Since over the real numbers." *Journal of Logic and Computation*, 1(2):229–259.

Gabbay, D. M. and P. McBrien [1991]. "Temporal Logic and Historical Databases." In *17th Conference on Very Large Databases*, pages 423–430, Barcelona.

Gabbay, D. M., I. M. Hodkinson, and M. A. Reynolds. "Temporal Logic — Mathematical Foundations and Computational Aspects." To appear in OUP, 1994.

Gabbay, D. M. [1981a]. "An Irreflexivity Lemma." In Monnich, U., editor, *Aspects of Philosophical Logic*, pages 67–89. Reidel, Dordrecht.

Gabbay, D. M. [1981b]. "Expressive Functional Completeness in Tense Logic." In Monnich, U., editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel, Dordrecht.

Gabbay, D. M. [1987]. "The Declarative Past and the Imperative Future." In Banieqbal, B. *et al.*, editors, *Coloquium on Temporal Logic and Specifications — Lecture Notes in Computer Science 389*, Manchester. Springer-Verlag.

Gabbay, D. M. [1990]. "Temporal Logic, Tense or Non-tense — inaugural lecture at Imperial College (17 May 1988)." In Spencer-Smith, R. and S. Torrance, editors, *Machinations — Computational Studies of Logic, Language and Cognition*. Ablex Publishing Co.

Gabbay, D. M. [1991a]. "Labelled Deductive Systems – Part I." Technical Report CIS–Bericht–90–22, Universität München, Centrum für Informations – und Sprachverarbeitung.

Gabbay, D. M. [1991b]. "Modal and Temporal Logic Programming III — metalevel features in the object level." In Cerro, L. F. and M. Penttonen, editors, *Non-Classical Logic Programming*. Oxford University Press.

Gabbay, D. M. [1991c]. "Theoretical Foundations for Non-monotonic Reasoning Part 2: Structured Non-monotonic Theories." In *SCAI 91 – Third Scandinavian Conference on AI*, pages 19–40.

Gabbay, D. M. "Fibred semantics and combinations of logics." Manuscript, Imperial College, 1992.

Garson, J. W. [1984]. "Quantification in Modal Logic." In Gabbay, D. and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 249–307. D. Reidel Publishing Company.

Gunadhi, H. and A. Segev [1993]. "Efficient Indexing Methods for Temporal Relations." *Transactions on Knowledge and Data Engineering*, 5(3):495–509.

Halpern, J. Y. and Y. Moses [1985]. "A Guide to the Modal Logics of Knowledge and Belief." In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 480–490.

Halpern, J. Y. and Y. Shoham [1986]. "A Propositional Modal Logic of Time Intervals." In *Proceedings of the Symposium on Logics in Computer Science – LICS86*, pages 279–292, Washington.

Heijennort, J., editor [1982]. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931.* Harvard University Press, Cambridge, Massachussets.

Hilbert, D. [1925]. "On the Infinite." In [Heijennort 1982], pages 367–392. Harvard University Press.

Hilbert, D. [1927]. "The Foundations of the Mathematics." In [Heijennort 1982], pages 464–479. Harvard University Press.

Hintikka, J. [1962]. *Knowledge and Belief.* Cornell University Press.

Hughes, G. E. and M. J. Cresswell [1968]. *An Introduction to Modal Logic.* Methuen.

Jensen, C. S., J. Clifford, S. K. Gadia, A. Segev, and R. T. Snodgrass [1992]. "A Glossary of Temporal Database Concepts." *SIGMOD RECORD*, 21(3):35–43.

Kabanza, F., J. M. Stevenne, and P. Wolper [1990]. "Handling Infinite Temporal Data." In *Proc. ACM Symposium on Principles of Database Systems*, pages 392–403.

Kamp, J. A. W. [1968]. *Tense Logic and the Theory of Linear Order.* PhD thesis, UCLA.

Kamp, J. A. W. [1971]. "Formal Properties of Now." *Theoria*, 35:227–273.

Konolige, K. [1986]. *A Deductive Model of Belief.* Research notes in Artificial Intelligence. Morgan Kaufmann.

Kowalski, R. A. and M. J. Sergot [1986]. "A Logic Based Calculus of Events." *New Generation Computing*, 4(1):67–95.

Kracht, M. and F. Wolter [1991]. "Properties of independently axiomatizable bimodal logics." *Journal of Symbolic Logic*, 56(4):1469–1485.

Kröger, F. [1987]. *Temporal Logics of Programs.* Springer EATCS Monographs on Theoretical Computer Science. Springer-Verlag.

Lloyd, J. W. [1987]. *Foundations of Logic Programming.* Springer-Verlag, second edition.

Loucopoulos, P., P. McBrien, U. Persson, F. Schumacker, and P. Vasey [1990]. "TEMPORA — Integrating Database Technology, Rule Based Systems and Temporal Reasoning for Effective Software." In European Communities, Commission, editor, *The Annual ESPRIT Conference*, pages 388–411, Brussels. Kluwer Academic Publishers.

Maier, D. [1983]. *The Theory of Relational Databases.* Pitman Publishing Limited.

Manning, K. J. and I. Torsun [1989]. "The Application of Temporal Logic to PAYE Tax Regulation." Technical report, Department of Computing, Bradford University.

McBrien, P., M. Niézette, D. Pantazis, A. Seltveit, U. Sundin, B. Theodoulidis, G. Tziallas, and R. Wohed [1991]. "A Rule Language to Capture and Model Business Policy Specifications." In *Proceedings of the Third Nordic Conference on Advanced Information Systems Engineering.* Springer-Verlag. LNCS 498.

McBrien, P. [1992]. "The Query and Updating of a Historical Database held in an RDMS." Technical report, Imperial College.

McKenzie, Jr., L. E. and R. T. Snodgrass [1991]. "Evaluation of Relational Algebra Incorporating the Time Dimension in Databases." *ACM Computing Surveys*, 23(4):501–544.

Morgenstern, M. [1983]. "Active Databases as a Paradigm for Enhanced Computing Environments." In *Ninth International Conference on Very Large Data Bases*, pages 34–42.

Navathe, S. B. and R. Ahmed [1988]. "TSQL — a language interface for history databases." In Rolland, C., F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 109–122. North Holland.

Pnueli, A. [1977]. "The Temporal Logic of Programs." In *Proc. 18th Symposium on the Foundations of Computer Science.*

Prior, A. [1957]. *Time and Modality.* Oxford University Press.

Quine, W. V. O. [1960]. *Word and Object.* MIT Press.

Ramakrishnan, R., F. Bancilhon, and A. Silberschatz [1987]. "Safety of Recursive Horn Clauses with Infinite Relations." In *Proc. Sixth Symposium on Principles of Database Systems*, pages 328–339.

Reiter, R. [1984]. "Towards a Logical Reconstruction of Relational Database Theory." In Brodie, M. L. *et al.*, editors, *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, pages 191–233. Springer-Verlag.

Reynolds, M. A. [1992]. "An Axiomatisation for Until and Since over the reals without the IRR Rule." *Studia Logica*, 51(2):165–194.

Segerberg, K. [1973]. "Two-dimensional Modal Logic." *J. of Philosophical Logic*, 2:77–96.

Sellis, T., C.-C. Lin, and L. Raschid [1993]. "Coupling Production Systems and Database Systems: A Homogeneous Approach." *IEEE Transaction on Knowledge and Data Engineering*, 5(2):240–256.

Snodgrass, R. and I. Ahn [1985]. "A Taxonomy of Time in Databases." In *ACM SIGMOD International Conference on Management of Data*, pages 236–246, Austin, Texas.

Sripada, S. M. [1990]. "A Basis for Historical Deductive Databases." Internal report, Imperial College, Department of Computing.

Stonebraker, M., E. Hanson, and S. Potamianos [1988]. "The POSTGRESS Rule Manager." *IEEE Transaction on Software Engineering*, 14(7):897–907.

Tansel, A., J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors [1993]. *Temporal Databases: Theory, Design and Implementaition.* Database Systems and Application Series. Benjamin/Cummings Pub. Co.

Thomason, S. K. [1980]. "Independent Propositional Modal Logics." *Studia Logica*, 39:143–144.

Thomason, R. H. [1984]. "Combinations of Tense and Modality." In Gabbay, D. and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 135–165. D. Reidel Publishing Company.

Tuzhilin, A. and J. Clifford [1990]. "A Temporal Relational Algebra as a Basis for Relational Completeness." In *Proc. 16th Conference on Very Large Databases*, Brisbane, Australia.

Ullman, J. D. [1988]. *Principles of Database and Knowledge-base systems*, volume I. Computer Science Press.

van Benthem, J. [1983]. *The Logic of Time*. Reidel, Dordrecht.

Venema, Y. [1990]. "Expressiveness and Completeness of an Interval Tense Logic." *Notre Dame Journal of Formal Logic*, 31(4).

Whitehead, A. N. and B. A. W. Russel [1910]. *Principia Mathematica*. Cambridge University Press.

Xu, M. [1988]. "On some U, S-Tense Logics." *Journal of Philosophical Logic*, 17:181–202.

Zaniolo, C. [1986]. "Safety and compilation of non-recursive horn clauses." In *Proc. First Int. Conf. on Expert Database Systems*, pages 237–252.