Formalizing planning algorithms: a logical framework for the research on extending the classical planning approach*

Silvio do Lago Pereira and Leliane Nunes de Barros
Institute of Mathematics and Statistics – University of São Paulo
Rua do Matao, 1010 – 05508-090 – Sao Paulo, SP
{slago,leliane}@ime.usp.br

Abstract

In this work we show how a planner implemented as an abductive reasoning process can have the same performance and behavior as classical planning algorithms. We demonstrate this result by considering three different versions of an abductive event calculus planner on reproducing some important comparative analyses of planning algorithms found in the literature. We argue that a logic-based planner, defined as the application of general purpose theorem proving techniques to a general purpose action formalism, can be a very solid base for the research on extending the classical planning approach.

Keywords: *abduction, event calculus, theorem proving, planning.*

Introduction

In general, in order to cope with domain requirements, any extension to STRIPS representation language would require the construction of complex planning algorithms, whose soundness can not be easily proved. The so called practical planners, which are said to be capable of solving complex planning problems, are constructed in an *ad hoc* fashion, making difficult to explain why they work or why they present a successful behavior. The main motivation for the construction of logic-based planners is the possibility to specify planning systems in terms of general theories of action and implement them as general purpose theorem provers, having a guarantee of soundness. Another advantage is that a planning system defined in this way has a close correspondence between specification and implementation. There are several works aiming the construction of sound and complete logic-based planning systems (Green 1969), (Eshghi 1988),(Missiaen, Bruynooghe, & Denecker 1994). More recent research results (Shanahan 2000) demonstrate that a good theorectical solution can coexist with a good practical solution, despite of contrary widespread belief (Russell & Norvig 2003).

In this work, we report on the implementation and analysis of three different versions of the AECP, a particular logic-based planner which uses *event calculus* (Shanahan 1995) as a formalism to reason about actions and change and *abduction* (Kakas, Kowalski, & Toni 1992) as an inference rule. By reproducing some important results on comparative analyses of planning algorithms (Knoblock & Yang 1994) (Kambhampati, Knoblock, & Yang 1995), and including experiments with the corresponding versions of the abductive event calculus planner, we show that there is a close correspondence between well known planning algorithms and logic-based planners. We also show that the efficiency results observed with a logic-based planner that adopts abductive event calculus and theorem proving can be comparable to that observed with some practical planners. We claim that one should start from an efficient logical implementation in order to make further extensions towards the specification of non-classical planners.

*This work has been supported by the Brazilian sponsoring agencies Capes, FAPESP and CNPq.

Abductive reasoning in the event calculus

Abduction is an inference principle that extends deduction, providing hypothetical reasoning. As originally introduced by (Peirce 1931 1958), it is an unsound inference rule that resembles a reverse modus ponens: if we observe a fact β, and we know $\alpha \rightarrow \beta$, then we can accept α as a possible explanation for β. Thus, abduction is a weak kind of inference in the sense that it only guarantees that the explanation is plausible, not that it is true.

Formally, given a set of sentences $\mathcal{B}$ describing a domain (*background theory*) and a sentence Γ describing an observation, the abduction process consists of finding a set of sentences Δ (*residue* or *explanation*) such that $\mathcal{B} \cup \Delta$ is consistent and $\mathcal{B} \cup \Delta \models \Gamma$. Clearly, depending on $\mathcal{B}$, for the same observed fact we can have multiple possible explanations. In general, the definition of *best explanation* depends on the context, but it is almost always related to some notion of minimallity. In practice, we should prefer explanations that postulates the minimum number of causes (Cox & Pietrzykowski 1986). Furthermore, abduction is, by definition, a kind of nonmonotonic reasoning, *i.e.* an explanation consistent, w.r.t. a determined knowledge state, can become inconsistent when new information is taken into account (Kakas, Kowalski, & Toni 1992).

Following, we present the event calculus as the formalism used to describe the background theory on planning domains and we show how the planning task can be understood as an

abductive process in the event calculus.

The event calculus formalism

The event calculus (Kowalski & Sergot 1986) is a formalism designed to model and reason about scenarios described as sets of events whose occurrences have the effect of starting or terminating the truth of determined properties (*fluents*) of the world. There are many versions of event calculi (Santos 1998). In this work, we use a version defined in (Shanahan 1995), whose axiomatization is the following:

$holdsAt(F,T) \leftarrow$ (EC1)
$\quad initially_p(F) \wedge \neg clipped(0,F,T)$

$holdsAt(F,T) \leftarrow$ (EC2)
$\quad happens(A,T_1,T_2) \wedge initiates(A,F,T_1) \wedge$
$\quad (T_2 \prec T) \wedge \neg clipped(T_1,F,T)$

$\neg holdsAt(F,T) \leftarrow$ (EC3)
$\quad initially_n(F) \wedge \neg declipped(0,F,T)$

$\neg holdsAt(F,T) \leftarrow$ (EC4)
$\quad happens(A,T_1,T_2) \wedge terminates(A,F,T_1) \wedge$
$\quad (T_2 \prec T) \wedge \neg declipped(T_1,F,T)$

$clipped(T_1,F,T_2) \leftrightarrow$ (EC5)
$\quad \exists A,T_3,T_4[happens(A,T_3,T_4) \wedge (T_1 \prec T_3) \wedge$
$\quad (T_4 \prec T_2) \wedge [terminates(A,F,T_3) \vee releases(A,F,T_3)]]$

$declipped(T_1,F,T_2) \leftrightarrow$ (EC6)
$\quad \exists A,T_3,T_4[happens(A,T_3,T_4) \wedge (T_1 \prec T_3) \wedge (T_4 \prec T_2) \wedge$
$\quad [initiates(A,F,T_3) \vee releases(A,F,T_3)]]$

$happens(A,T_1,T_2) \rightarrow T_1 \leq T_2$ (EC7)

Table 1: *Predicates of the event calculus*

predicate	*description*
$holdsAt(F,T)$	fluent F holds at time T
$initially_p(F)$	fluent F holds from time 0
$initially_n(F)$	fluent F does not hold from time 0
$happens(A,T_1,T2)$	event A starts at T_1 and finishes at T_2
$initiates(A,F,T)$	event A starts fluent F at time T
$terminates(A,F,T)$	event A finishs fluent F at time T
$releases(A,F,T)$	event A releases fluent F at time T
$clipped(T_1,F,T_2)$	fluent F ceased to hold in $[T_1,T_2]$
$declipped(T_1,F,T_2)$	fluent F started to hold in $[T_1,T_2]$

In the event calculus, the frame problem is overcome through circumscription. Given Σ a domain description expressed as a conjunction of formulae that does not include the predicates $initially$ or $happens$; Δ a narrative of actions expressed as a conjunctions of formulae that does not include the predicates $initiates$, $terminates$ or $releases$; Ω a conjunction of uniqueness-of-names axioms for actions and fluents; and EC a conjunction of the axioms of the event calculus, we have to consider the following formula as the background theory on the abductive event calculus planning:

$$\text{CIRC}[\Sigma; initiates, terminates, releases] \wedge$$
$$\text{CIRC}[\Delta; happens] \wedge \Omega \wedge EC,$$

where $\text{CIRC}[\Sigma; \rho_1, \ldots, \rho_n]$ means the circumscription of Σ with relation to the predicate symbols $\rho_1, \ldots, \rho_n$. By circumscribing $initiates$, $terminates$ and $releases$ we are imposing that the known effects of actions are the only effects of actions, and by circumscribing $happens$ we assume that there are no unexpected event occurrences. An extended discussion about the frame problem and its solution through circumscription can be found in (Shanahan 1997).

An abductive event calculus planner

Planning in the event calculus is naturally handled as an abduction process (Eshghi 1988). In this setting, given a domain description Σ, the task of planning a sequence of actions Δ in order to satisfy a given goal Γ corresponds to an abductive process expressed by:

$$\text{CIRC}[\Sigma; initiates, terminates, releases] \wedge$$
$$\text{CIRC}[\Delta; happens] \wedge \Omega \wedge EC \models \Gamma,$$

where Δ – *the abductive explanation* – is a plan for the goal Γ. In (Shanahan 2000) a planning system based on this idea is presented as a PROLOG abductive meta-interpreter. This meta-interpreter is specialized for the event calculus by compiling the EC axioms into its meta-clauses. The main advantage of this compilation is to allow an extra level of control to the planner. In particular, it allows us to define an ordering in which subgoals can be achieved, improving efficiency and giving special treatment to predicates that represent incomplete information. By incomplete information we mean predicates for which we do not assume its closure, *i.e.* we can not use negation as failure to prove their negations, since they can be abduced. The solution to this problem is to give a special treatment for negated literals with incomplete information at the meta-level. In the case of partial order planning, we have incomplete information about the predicate $before$, allowing the representation of partial order plans. Thus, when the meta-interpreter finds a literal $\neg before(X,Y)$, it tries to prove it by adding $before(Y,X)$ to the plan (*abductive residue*) and checking its consistence.

In the abductive event calculus planner – AECP – a planning problem is given by a *domain* description represented by a set of clauses $initiates$, $terminates$ and $releases$, an *initial state* description represented by a set of clauses $initially_p$ and $initially_n$, and a *goal* description represented by a list of literals $holdsAt$. As solution, the planner returns an abductive residue composed by literals $happens$ and $before$ (the partial order plan) and a negative residue composed of literals $clipped$ and $declipped$ (the causal links of the partial order plan).

Classical planning in the event calculus

In order to perform a fair comparative analysis with STRIPS-like planning algorithms, some modifications have to be done in the AECP, which are related to the following assumptions in classical planning: (*i*) atomic time, (*ii*) deterministic effects and (*iii*) omniscience. From (*i*) follows that we need to change the predicate $happens(A,T1,T2)$ to a binary version. Thus, $happens(A,T)$ means that the action A happens at time T and, by doing this change, the axiom EC7 will be no longer necessary. From (*ii*) follows that there is no need for the predicate $releases$ and, finally, from (*iii*) (remembering the fact that STRIPS's action representation does not allow negative preconditions), follows that there is

no need for predicate $initially_n$ neither the axioms EC3, EC4 and EC6. With these changes, we can specify a simplified axiomatization of the event calculus containing only its relevant aspects to the classical planning:

$holdsAt(F,T) \leftarrow initially(F) \wedge \neg clipped(0,F,T)$ (SEC1)

$holdsAt(F,T) \leftarrow$
$happens(A,T_1) \wedge initiates(A,F,T_1) \wedge (T_1 \prec T) \wedge$
$\neg clipped(T_1,F,T)$ (SEC2)

$clipped(T_1,F,T_2) \leftrightarrow \exists A,T[happens(A,T) \wedge (T_1 \prec T) \wedge (T \prec T_2) \wedge$
$terminates(A,F,T)]$ (SEC3)

The ABP planning system

Based on this simplified axiomatization, we have implemented the ABP planning system. This planner uses *iterative deepening search* (IDS) and *first-in, first-out* (FIFO) goal ordering, while AECP uses *depth first search* (DFS) and *last-in, last-out* (LIFO) strategies. Using IDS, we turn out the method complete and we increase the possibility to find minimal explanations. It is important to notice that in the original version of the AECP, both properties did not hold. Following, we explain the details of the knowledge representation and control knowledge decisions made in our implementations that are relevant on the comparative analysis presented in the next section.

Action representation. In the event calculus, the predicates $initiates$ and $terminates$ are used to describe the effects of an action. For instance, consider the predicate $walk(X,Y)$ representing the act of walking from x to y. The effects of this action can be described as:

$initiates(walk(X,Y),at(Y),T) \leftarrow$
$holdsAt(at(X),T) \wedge X \neq Y$
$terminates(walk(X,Y),at(X),T) \leftarrow$
$holdsAt(at(X),T) \wedge X \neq Y$

In the AECP's meta-level, the above clauses are represented by the predicate $axiom(H,B)$, where H is the head of the clause and B is its body, that is:

```
axiom(initiates(walk(X,Y),at(Y),T),
  [holdsAt(at(X),T),X\=Y]).
axiom(terminates(walk(X,Y),at(X),T),
  [holdsAt(at(X),T),X\=Y]).
```

Similarly, the STRIPS representation of this action is:

```
oper(walk(X,Y),[at(X),X\=Y],[at(Y)],[at(X)]).
```

Note that, in the STRIPS representation, the first parameter of the predicate `oper` is the action's name, while in the EC representation, the first parameter of the predicate `axiom` is `initiates` or `terminates`. Since PROLOG's indexing method uses the first parameter as the searching key, finding an action with the predicate `oper` would take constant time, while a search with the predicate `axiom` would take time proportional to the number of clauses for this predicate included in the knowledge base. Thus, in order to establish a suitable correspondence between both approaches, in the implementation of the ABP, the clauses of the form `axiom(initiates(`α,ϕ,T`),`B`)` are represented at the meta-level as `initiates(`α,ϕ,T,B`)`. In analogous way, the clauses `axiom(terminates(`α,ϕ,T`),`B`)` are represented as `terminates(`α,ϕ,T,B`)`.

Abducible and executable predicates. In the AECP (Shanahan 2000), the meta-predicates `abducible` and `executable` are used to establish which are the abducible predicates and the executable actions, repectively. The declaration of the abducible predicates is important to the planner, as it needs to know the predicates with incomplete information that can be added to the residue. By restricting the facts that can be abduced, we make sure that only basic explanations are computed (*i.e.* those explanations that can not be formulated in terms of others effects). On the other hand, the declaration of executable actions only makes sense in HTN planners, where it is important to distinguish between primitive and compound actions. Since in this work we only want to compare the logical planner with partial order planners, we can assume that all the actions in the knowledge base are executable and that the only abducible predicates are `happens` and `before` (the same assumption is made in the STRIPS-like partial order planners).

Codesignation constraints. Since the AECP uses the PROLOG's unification procedure as the method to add codesignations constraints to the plan, it is difficult to compare it with the STRIPS-like planning algorithms (which have a special procedure implemented for this purpose). So, we have implemented ABP as a propositional planner, as is commonly done in most of the performance analyses in the planning literature. As we will see, this change has positively affected the verification of the consistency of the negative residue.

Consistency of the negative residue. In the AECP, the negative residue has to be checked for consistency every time the positive residue H is modified. This behavior corresponds to an interval protection strategy for the predicate `clipped` (in a way equivalent to *book-keeping* in partial order planning). However, in the case of a propositional planner, we have only to check for consistency a new literal `clipped` (added to the negative residue) with respect to the actions already in the positive residue, and a new literal `happens` (added to the positive residue) with respect to the intervals already in the negative residue. Thus, in contrast with the performance presented by the AECP, the conflict treatment in the ABP is incremental and has a time complexity of $O(|H|)$. In addition, when an action in the plan is selected as the establisher of a subgoal, only the new added literal `clipped` has to be protected.

Systematicity and redundancy

In order to analyse the performance of the abductive event calculus planner, we have implemented three different planning strategies:

- ABP: abductive planner (equivalent to POP);
- SABP: systematic version of ABP (equivalent to SNLP);
- RABP: redundant version of ABP (equivalent to TWEAK).

Systematicity. A systematic version of the ABP, called SABP, can be obtained by modifying the event calculus axiom SEC3 to consider as a "threat" to a fluent F not only an action that terminates it, but also an action that initiates it:

$clipped(T_1, F, T_2) \leftrightarrow \exists A, T[happens(A, T) \wedge (T_1 \prec T) \wedge (T \prec T_2) \wedge$ (SEC3')

$[terminates(A, F, T) \vee initiates(A, F, T)]]$

With this simple change, we expect that SABP will have the same performance of systematic planners, like SNLP (MacAllester & Rosenblitt 1991), and the same trade-off performance with the corresponding redundant version of the ABP planner.

Redundancy. A redundant version of the ABP, called RABP, does not require any modification in the EC axioms. The only change that we have to make is in the goal selection strategy. In the ABP, as well in the SABP, subgoals are selected and then eliminated from the list of subgoals as soon as they are satisfied. This can be safely done because those planners apply a causal link protection strategy.

A MTC – *modal truth criterion* – strategy for goal selection can be easily implemented in the RABP by performing a temporal projection. This is done by making the meta-interpreter to "execute" the current plan, without allowing any modification on it. This process returns as output the first subgoal which is not necessarily true.

Another modification is on the negative residue: the RABP does not need to check the consistency of negative residues every time the plan has been modified. So, in the RABP, the negative literals of the predicate `clipped` does not have a special treatment by the meta-interpreter. As in TWEAK (Chapman 1987), this will make the RABP to select the same subgoal more than once but, on the other hand, it can plan for more than one subgoal with a single goal establishment.

The comparative analysis

In order to show the correspondence between abductive planning and partial order planning, we have implemented the abductive planners (ABP, SABP and RABP) and three well known partial order planning algorithms (POP, SNLP and TWEAK). All these planners have been implemented in PROLOG and all the cares necessary to guarantee the validity of the comparisons have been taken (*e.g.* all the planners shared common data structures and procedures). A complete analysis of these results is presented in (Pereira & Barros 2001) and (Pereira 2002).

We have performed two experiments with these six planners: (*i*) evaluation of the correspondence between abductive planning in the event calculus and partial order planning and (*ii*) evaluation of systematicity/redundancy obtained with different goal protection strategies.

Experiment I: correspondence between POP and ABP

In order to evaluate the relative performance of the planners POP and ABP, we have used the artificial domains family $D^m S^n$ (Barrett & Weld 1994). With this, we ensure that the empirical results we have obtained were independent of the idiosyncrasies of a particular domain.

Based on these domains, we have performed two tests: in the first, we observe how the size of the search space explored by the systems increases as we increase the number of subgoals in the problems; in the second, we observe how the average CPU-time consumed by the systems increases as we increase the number of subgoals in the problems.

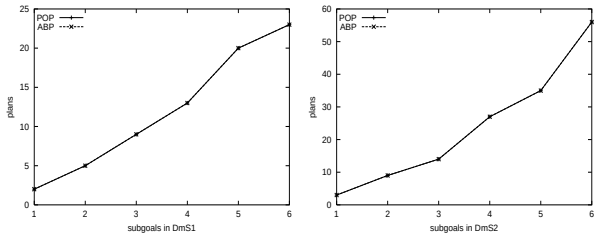

Figure 1: Search space size to solve problems in $D^m S^n$

In figure 1, we can observe that the ABP and POP explore identical search spaces. Therefore, we can conclude that they implement the same planning strategies (*i.e.* they examine the same number of plans, independently of the fact that they implement different approaches). This result extends the work presented in (Shanahan 2000), which verifies the correspondence between abductive planning in the event calculus (AECP) and partial order planning (POP) only in an informal way, by inspecting the code. In figure 2, we can observe that, for all problems solved, the average CPU-time consumed by both planners is approximately the same. This proves that the necessary inferences in the logical planners do not increase the time complexity of the planning task.

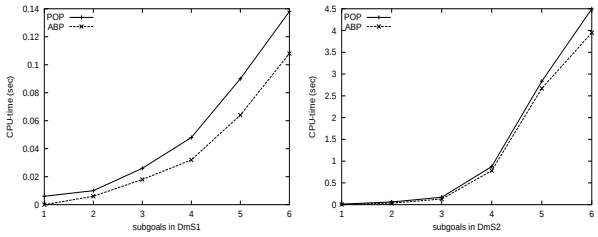

Figure 2: Average CPU-time to solve problems in $D^m S^n$

Therefore, through this first experiment, we have corroborated the conjecture that abductive planning in the event calculus is isomorphic to partial order planning (Shanahan 2000). Also, we have showed that, using abduction as inference rule and event calculus as formalism to reasoning about actions and chance, a logical planning system can be as efficient as a partial order planning system, with the advantage that its specification is "directly executable".

Experiment II: trade-off between systematicity and redundancy

There was a belief that by decreasing redundancy it would be possible to improve planning efficiency. So, a systematic planner, which never visits the same plan twice in its search space, would be more efficient than a redundant planner (MacAllester & Rosenblitt 1991). However, (Kambhampati 1993) has shown that there is a trade-off between redundancy elimination and least commitment: redundancy is eliminated at the expense of increasing commitment in the planner. Therefore, the performance of a partial order planner is better predicted based on the way it deals with the trade-off between redundancy and commitment than on the systematicity of its search.

In order to show the effects of this trade-off, Kambhampati chose two well known planning algorithms: TWEAK and SNLP.TWEAK does not keep track of which goals were already achieved and which remains to be achieved. Therefore, TWEAK may achieve and clobber a subgoal arbitrarily many times, having a lot of redundancy on its search space. On the other hand, SNLP achieves systematicity by keeping track of the causal links of the plans generated during search, and ensuring that each branch of the search space commits to and protects mutually exclusive causal links for the partial plans, *i.e.* it protects already established goals from negative or positive threats. Such protection corresponds to a strong form of premature commitment (by imposing ordering constraints on positive threats) which can increase the amount of backtracking as well as the solution depth, having an adverse effect on the performance of the planner.

Kambhampati's experimental analyses show that there is a spectrum of solutions to the trade-off between redundancy and commitment in partial order planning, in which the SNLP and TWEAK planners fall into opposite extremes. To confirm this result, and to show that it is valid to abductive planners too, we created a new family of artificial domains, called $A^x D^y S^2$ (Pereira 2002), through which we can accurately control the ratio between the number of positive threats x (*i.e.* distinct actions that contribute with one same effect) and negative threats y (*i.e.* distinct actions that contribute with opposing effects) in each domain. To observe the behavior of the compared planners, as we vary the ratio between the number of positive and negative threats in the domains, we keep constant the number of subgoals in the solved problems. Then, as a consequence of this fact and of the characteristics of the domains in the family $A^x D^y S^2$, the number of steps in all solutions stays always the same.

The results of this second experiment (figure 3), show that the systematic and redundant versions of the abductive planner (SABP and RABP) have the same behavior of its corresponding algorithmic planners (SNLP and TWEAK).

So, we have extended the results of the previous experiment and show that the isomorphism between abductive reasoning in the event calculus and partial order planning can be preserved for systematic and redundant methods of planning. Moreover, we also corroborate the conjecture that the performance of a systematic or redundant planner is strongly related to the ratio between the number of positive and negative threats in the considered domain (Knoblock & Yang 1994) and that this conjecture remains valid to abductive planning in the event calculus.

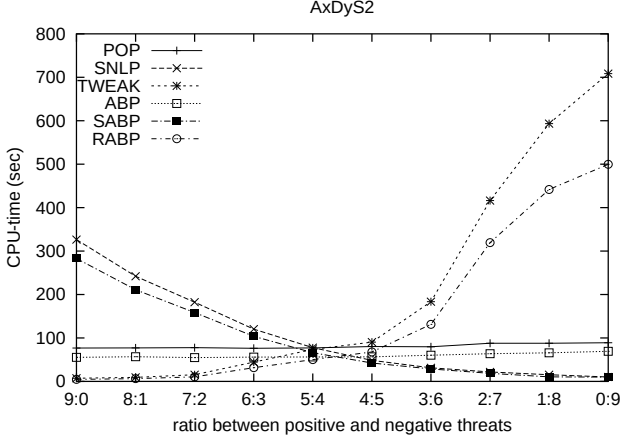

Figure 3: Average CPU-time to solve problems in $A^x D^y S^2$

Conclusion

The main contribution of this work is: (*i*) to propose a formal specification of different well-known algorithms of classical planning and (*ii*) to show how a planner based on theorem proving can have similar behavior and performance to those observed in partial order planners based on the STRIPS. One extra advantage of our formal specification is its close relationship with a PROLOG implementation, which can provide a good framework to test extensions to the classical approach, as well to the integration of knowledge-based approaches for planning.

It is important to notice that the original version of the AECP proposed by (Shanahan 2000) does not guarantee completeness neither minimal plan solution. However, the abductive planners we have specified and implemented guarantee these properties by using IDS (iterative deepening search) and FIFO goal ordering strategies.

We are currently working on the idea proposed in (Barros & Pereira 2002) which aims to build, on the top of our abductive planners, a high-level robot programming language for applications in cognitive robotics. First, we have implemented a HTN version of the abductive event calculus planner to cope with the idea of high-level specifications of robotic tasks. Further, we intend to work with planning and execution with incomplete information.

References

Barrett, A., and Weld, D. S. 1994. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence* 67(1):71–112.

Barros, L. N., and Pereira, S. L. 2002. High-level robot programs based on abductive event calculus. In *Proceedings of 3rd International Cognitive Robotics Workshop*.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3):333–377.

Cox, P. T., and Pietrzykowski, T. 1986. Causes for events: their computation and applications. In *Proc. of the 8th international conference on Automated deduction*, 608–621. Springer-Verlag New York, Inc.

Eshghi, K. 1988. Abductive planning with event calculus. In *Proc.of the 5th International Conference on Logic Programming*. 562–579.

Green, C. 1969. Application of theorem proving to problem solving. In *IJCAI*. 219–240.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1992. Abductive logic programming. *Journal of Logic and Computation* 2(6):719–770.

Kambhampati, S.; Knoblock, C. A.; and Yang, Q. 1995. Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* 76(1-2):167–238.

Kambhampati, S. 1993. On the utility of systematicity: Understanding tradeoffs between redundancy and commitment in partial-ordering planning. In *Foundations of Automatic Planning: The Classical Approach and Beyond: Papers from the 1993 AAAI Spring Symposium*, 67–72. AAAI Press, Menlo Park, California.

Knoblock, C., and Yang, Q. 1994. Evaluating the tradeoffs in partial-order planning algorithms.

Kowalski, R. A., and Sergot, M. J. 1986. A logic-based calculus of events. In *New Generation Computing 4*. 67–95.

MacAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. 9th National Conference on Artificial Intelligence*. 634–639.

Missiaen, L.; Bruynooghe, M.; and Denecker, M. 1994. Chica, an abductive planning system based on event calculus.

Peirce, C. S. 1931-1958. *Collected Papers of Charles Sanders Peirce*. Harvard University Press.

Pereira, S. L., and Barros, L. N. 2001. Efficiency in abductive planning. In *Proceedings of 2nd Congress of Logic Applied to Technology*. 213–222.

Pereira, S. L. 2002. *Abductive Planning in the Event Calculus*. Master Thesis, Institute of Mathematics and Statistics - University of Sao Paulo.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach (second edition)*. Prentice-Hall, Englewood Cliffs, NJ.

Santos, P. E. 1998. Formalising the common sense of a mobile robot.

Shanahan, M. 1995. A circumscriptive calculus of events. *Artificial Intelligence* 77(2):249–284.

Shanahan, M. P. 1997. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press.

Shanahan, M. P. 2000. An abductive event calculus planner. In *The Journal of Logic Programming*. 44:207–239.