Business Process Management Using Process Algebra and Relational Database Model

Kelly Rosa Braghetto (kellyrb@ime.usp.br)[1]
João Eduardo Ferreira (jef@ime.usp.br)[1]
Calton Pu (calton@cc.gatech.edu)[2]

[1] Department of Computer Science
Institute of Mathematics and Statistics (IME)
University of São Paulo (USP)

[2] College of Computing
Georgia Institute of Technology

Agenda

Agenda

Agenda

Agenda

Business Process Management (BPM)

BPM involves methods, techniques and tools to support the entire business process life cycle:

- project;
- execution and management;
- operational analysis.

⇒ Associating formal frameworks to the project phase is valuable since:

- they provide non-ambiguous models;
- they improve the diagnosis capability;
- they enable a reliable execution control of the processes.

Business Process Management (BPM)

Formal Specification of Processes

Well-known examples of frameworks for formal reasoning about processes:

- Petri Nets (Place/Transitions-Nets, Coloured Petri Nets, Workflow Nets, ...);
- Process Algebras (Algebra of Communicating Processes, π-Calculus, LOTOS, ...).

$\Rightarrow$ There are tools based on formal frameworks for the management of workflows and business processes, but integrating these tools with other applications is not an easy task.

Our work

NavigationPlanTool (NPTool)

A tool that supports the design, instantiation and execution control of business processes supported by the process algebra formalism.

⇒ It uses *Navigation Plan Definition Language* and a relational database to specify the processes and to control their instantiations and executions.

Navigation Plan Definition Language (NPDL)

NPDL is a business process specification language based on *Algebra of Communicating Processes* operators that applies the concept of *Navigation Plan*, that belongs to *RiverFish* architecture.

$\Rightarrow$ NPTool implements NPDL as an extension of SQL language.

Navigation Plan Definition Language (NPDL)

Main Features

- Specifies processes by algebraic expressions (operators and atomic actions);
- Contains basic operators (from process algebra):

$$+, \cdot \text{ and } \|$$

- Contains additional operators (only in NPDL):

$$\%, \%!, |*, \&, \wedge \text{ and } ?$$

- Facilitates the representation of control-flow patterns and compensates for some limitations of process algebras and Petri Nets.

Navigation Plan Definition Language (NPDL)
Example

A simple calculation process

```
CREATE ACTION A1 'ReadFirstValue';
CREATE ACTION A2 'ReadSecondValue';
CREATE ACTION A3 'CalculateSum';
CREATE ACTION A4 'CalculateProduct';
CREATE ACTION A5 'ShowResult';
CREATE PROCESS P1 'CalculationProcessAux';
CREATE PROCESS P2 'CalculationProcess';
SET P1 = (A1 || A2).( A3 + A4 ).A5;
SET P2 = P1.P2 + P1;
```

Navigation Plan Definition Language (NPDL)
Other Commands

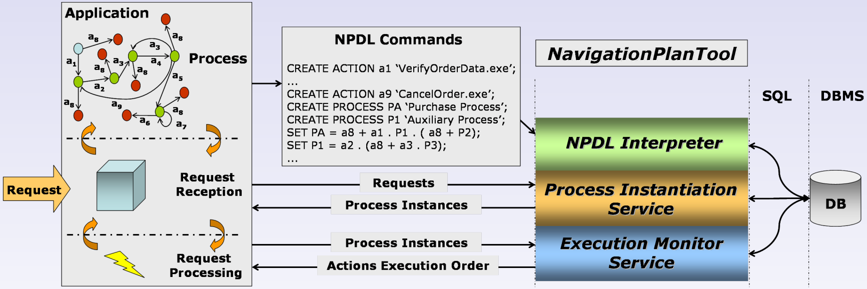
```
CREATE RULE [RULE NAME];
CREATE FUNCTION [FUNCTION NAME];
DROP PROCESS [PROCESS NAME];
DROP ACTION [ACTION NAME];
DROP RULE [RULE NAME];
DROP FUNCTION [FUNCTION NAME];
SELECT ACTIONS;
SELECT PROCESSES;
SELECT RULES;
SELECT FUNCTIONS;
...
```

NavigationPlanTool (NPTool)

NPTool is a library of functions implemented in Java that offers three important services:

1. NPDL Interpreter
2. Process Instantiation Service
3. Process Instance Execution Monitor

NavigationPlanTool (NPTool)
Services



NPDL Commands

```
CREATE ACTION a1 'VerifyOrderData.exe';
...
CREATE ACTION a9 'CancelOrder.exe';
CREATE PROCESS PA 'Purchase Process';
CREATE PROCESS P1 'Auxiliary Process';
SET PA = a8 + a1 . P1 . ( a8 + P2);
SET P1 = a2 . (a8 + a3 . P3);
...
```

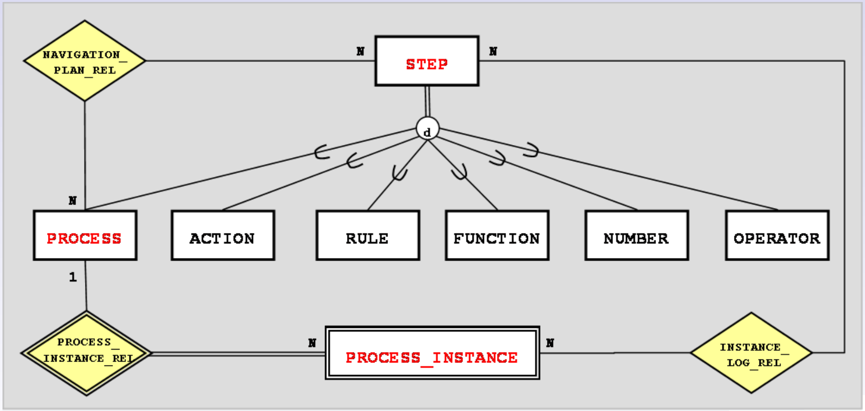
NPDL Interpreter

Operations:

1. Creation of relational data structures (tables) to store processes, actions and instances data;
2. Lexical, syntactic and semantical analysis of NPDL commands;
3. Translation of NPDL commands to “pure” SQL commands.

NPDL Interpreter

Relational Database Model



Process Instantiation

Following the concept of *navigation plan instantiation* of *Riverfish* architecture, this service offers functions for the creation of process instances.

⇒ A process instance represents a request to a specific process.

⇒ All the instance data, as well as process definition data associated with the instance, are stored in the database.

Process Instance Execution Monitor

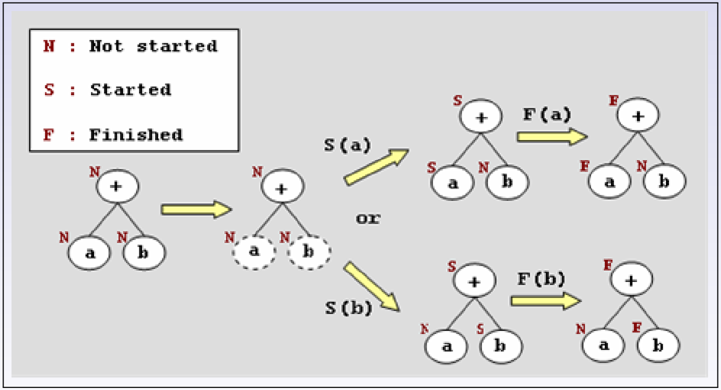
This service is responsible for linking a process instance to its execution data. It contains the functions that control the execution of a process instance.

The service is supported by database structures to store and recover the execution state of the instance and main-memory structures to control the execution flow.

$\Rightarrow$ Expression tree of the process + execution state of an instance = *navigation tree* of the instance.

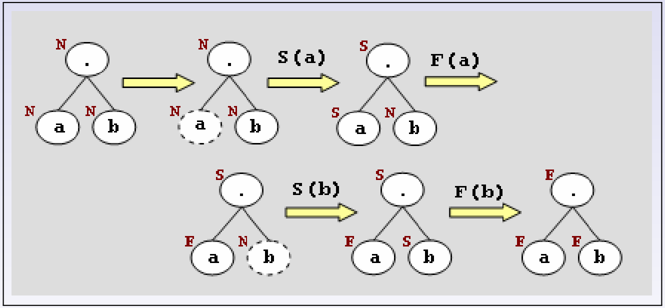
Navigation Trees

Treatment of basic operators: alternative composition $a + b$

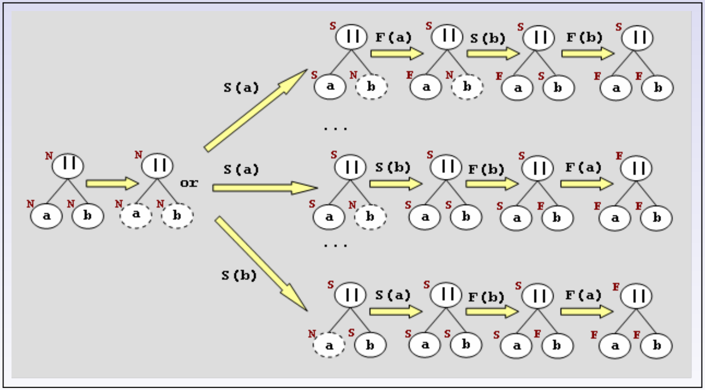


Navigation Trees

Treatment of basic operators: sequence composition $a \cdot b$

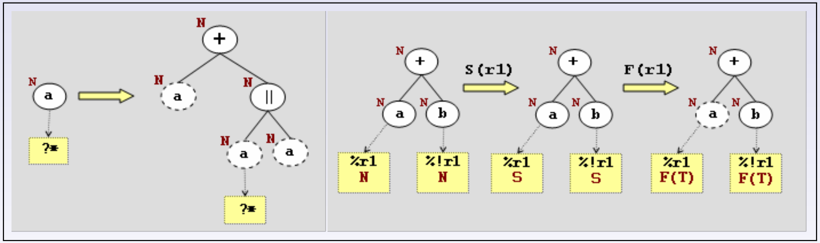


Navigation Trees
Treatment of basic operators: parallel composition $a \| b$



Navigation Trees

Treatment of additional operators:
- unlimited repetition $a?*$
- conditional execution $\%r_1 a + \%!r_1 b$



Example of Execution Control

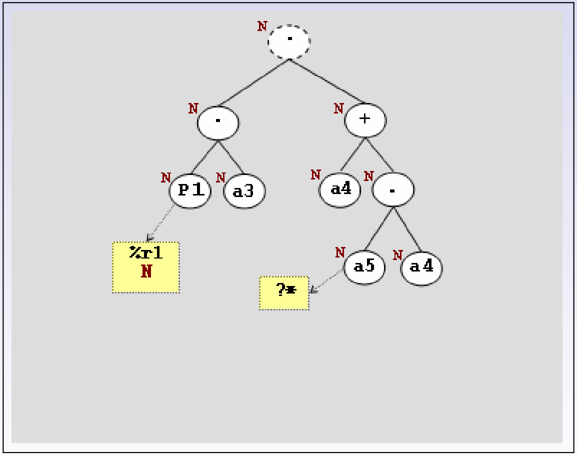
Acquisition of items in a library collection (simplified version)

- r_1 checks the completeness of data from acquisition order and the availability of the budget for purchasing;
- a_1 sorts the order items according to some priority;
- a_2 gathers the prices of the order items;
- a_3 liberates the order for purchasing;
- a_4 registers the purchase receiving;
- a_5 registers a problem with the purchase.

```
SET P1 = a1 || a2 + (a1 || a2).P1;
SET P = %r1 P1 . a3 . (a4 + a5?* . a4);
```

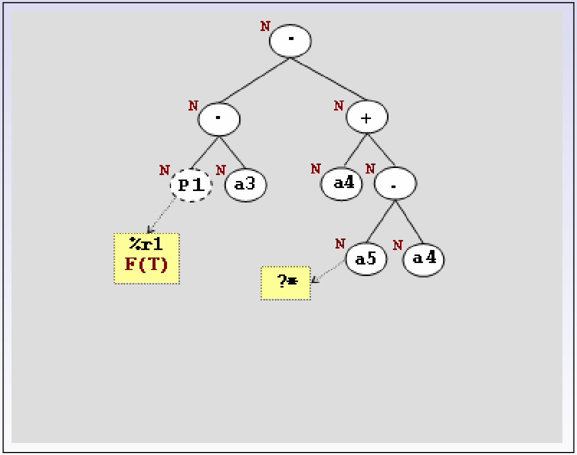
Navigation tree of an instance of acquisition process

Initial state



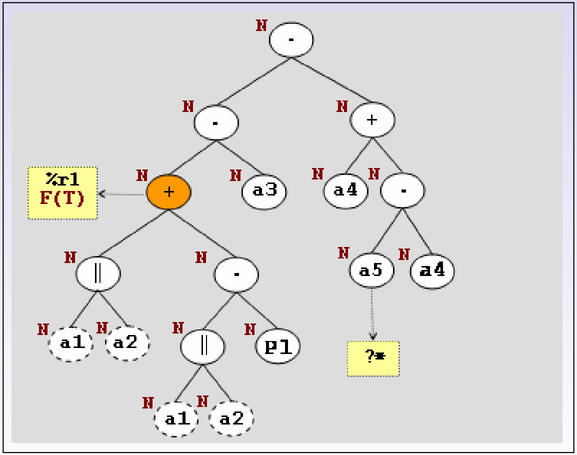
Navigation tree of an instance of acquisition process

After the execution of r_1 - check the completeness of order data



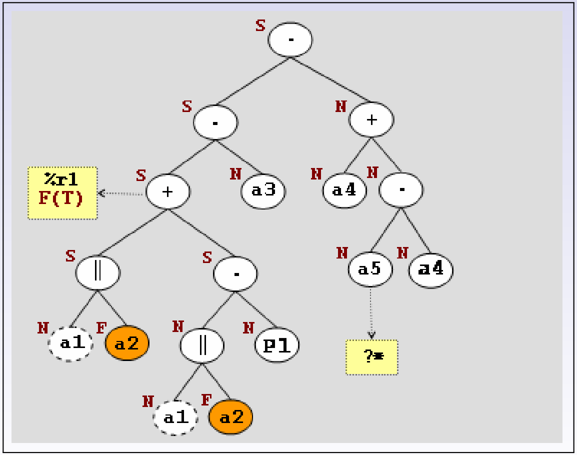
Navigation tree of an instance of acquisition process

After the substitution of P_1 - the subprocess for sort and gather the prices of order items



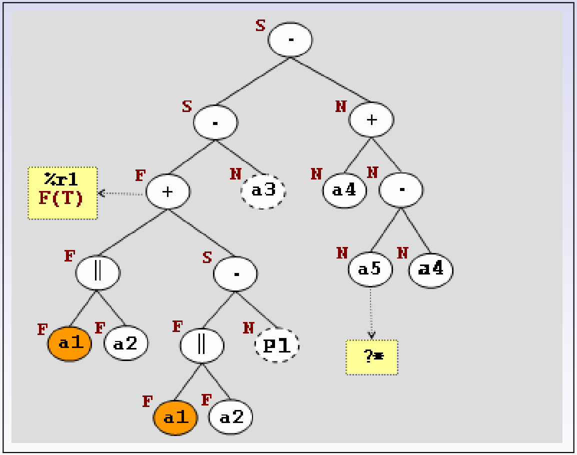
Navigation tree of an instance of acquisition process

After the execution of a_2 - gather the prices of other items



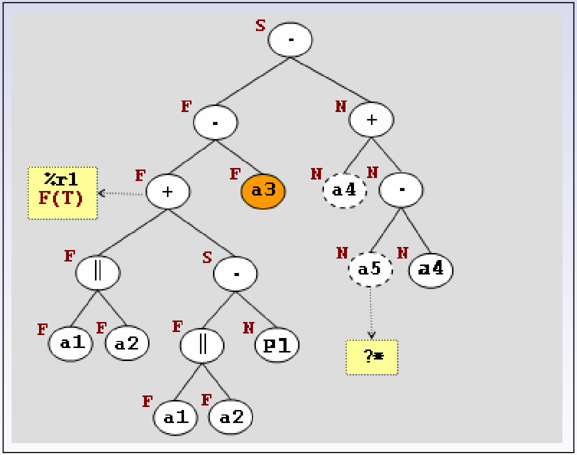
Navigation tree of an instance of acquisition process

After the execution of a_1 - sort the order prices according to some priority



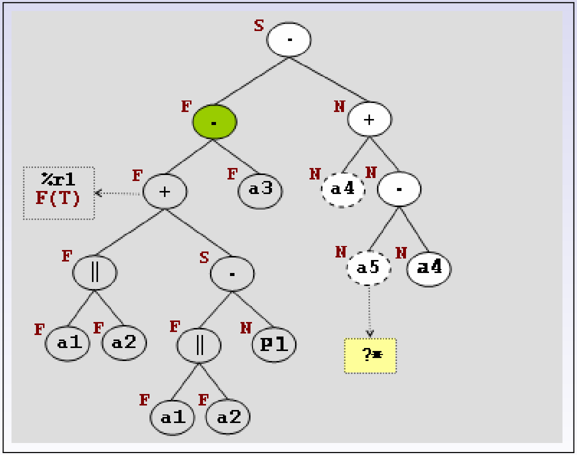
Navigation tree of an instance of acquisition process

After the execution of a_3 - liberates the order for purchasing



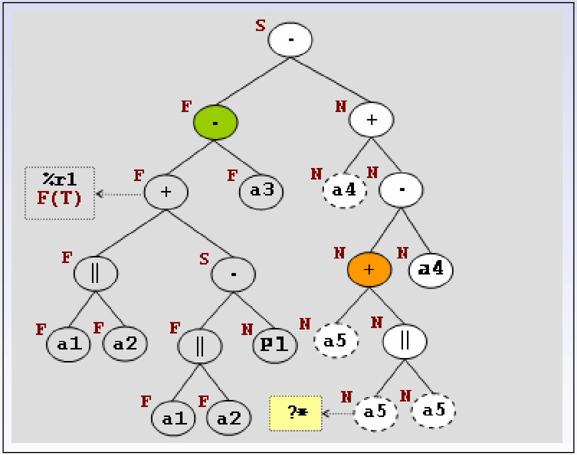
Navigation tree of an instance of acquisition process

After the removal of inaccessible branches



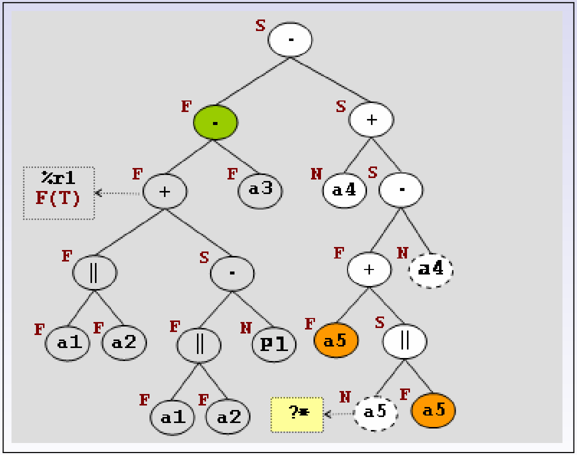
Navigation tree of an instance of acquisition process

After mapping operator ?*



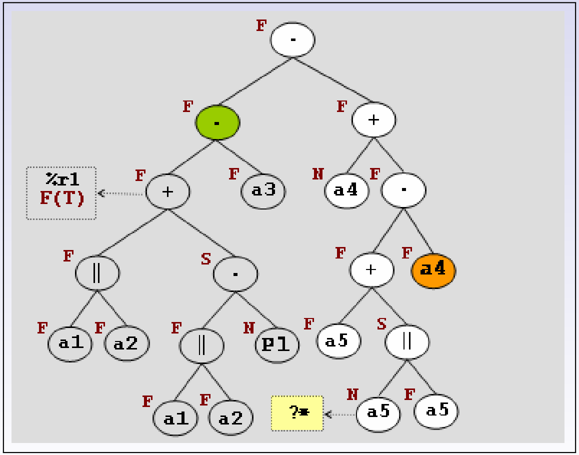
Navigation tree of an instance of acquisition process

After the execution of a_5 - register a problem with the purchase



Navigation tree of an instance of acquisition process

After the execution of a_4 - register the purchase receiving



Conclusion

Using process algebra as formal basis supported:

- the creation of a simple but flexible relational database model to represent processes data and a language to manipulate these data - the NPDL;
- the development of a reliable engine that uses the process algebra axioms and operational semantics to implement a execution control of business processes.

Conclusion

The database can be viewed as a common repository of processes.

- This approach allows us to share processes definitions between different applications that use the NPTool.
- The compositional characteristic of process algebra makes possible the composition of great processes from smaller ones.

Conclusion

Work in Progress

- Automated generation of NPDL expressions from graphical representations.
- NPDL extension to represent process data flow.
- Process mining based on statistical approach.