An Extension of Situation Calculus Applied to Project Management

Fabrício Jailson Barth and Edson Satoshi Gomi

Intelligent Techniques Laboratory
Department of Computer Engineering
Polytechnic School. University of São Paulo
Av. Prof. Luciano Gualberto, 158 tv. 3. 05508-900. São Paulo, SP, Brazil
{fabricio.barth,edson.gomi}@poli.usp.br

Abstract. The Situation Calculus has been used as a tool to develop planning systems. However, the traditional Situation Calculus does not supply mechanisms to represent actions that consume resources and represent time, which are features that are important in applications like Project Management. This work presents an extension of Situation Calculus which allows representation of time description and actions that consume human and material resources. The application of this extended calculus is exemplified by a project planning case of the telecommunications area.

1 Introduction

An intelligent agent must be capable to plan its actions in advance. Such ability is essential to achieve intelligent behavior and its implementation is extremely important in practical applications, e.g. project management, robotics, manufacture, logistic, space missions, etc.

Several planning algorithms had been proposed in the last years. In those proven correct there are many limitations, according to representation of actions and computational performance. Consequently, they cannot be applied to solve problems of the real world. On the other hand, practical planners, capable of solving many problems, have been constructed in ad-hoc way, being difficult to customize to be used in different applications.

An example of practical application is in management of projects. The domain of project management demands, among others features, the ability of to represent concurrent actions, resources usage and actions duration.

Through the use of formal logic it is possible to construct correct project plans based on well known principles and that can be validated, maintained and modified easily. It remains to be investigated, whether this kind of planners can be executed in a real domain, or can represent all relevant problem information.

McCarthy and Hayes [7] proposed a language based on logic first-order, named Situation Calculus. There are some restrictions of the Situation Calculus that obstruct the representation of actions of a practical domain, for example,

there is no easy way to represent actions that consume resources and represent duration.

This work presents an extension of Situation Calculus capable of representing actions that consume resources and explicit time. The goal is to apply this extended calculus to develop support tools for project management planning and controlling processes.

This paper is organised as follow. Section 2 presents some fundamental project management concepts. Section 3 presents the Situation Calculus. Section 4 describes how to associate a situation with a time point. Section 5 shows a proposal to represent actions that consume resources in the Situation Calculus. Section 6 shows how it is possible to represent resources expenses over time. Section 7 presents a logic program based on the extended Situation Calculus. Section 8 shows an example of implementation in project management domain. Finally, section 9 presents the final remarks.

2 Project Management

Organizations usually perform works that involves projects [3]. Different projects share many features. For example, they are performed by people, constrained by limited resources, and planned, executed and controlled.

A project can be defined as *"a project is a temporary endeavor undertaken to create a unique product or service"* [3]. *Temporary* means that every project there is a definite beginning and a definite end. *Unique* means that the product or service is different in some distinguishing way from all similar products or services.

Projects are undertaken at all levels of the organization. They may involve a single person or thousands of people. Projects may involve a single unit of one organization or may cross the limits of the organization. They are often critical components of the performing organization's business strategy. Examples of projects include: development of a new product or service, making a change in structure of an organization, designing a new transportation vehicle, etc.

Project Management is the application of knowledge, skills, tools, and techniques to project activities in order to find or exceed stakeholder needs and find expectations from a project. Finding or exceeding stakeholder needs and expectations invariably involves balancing competing demands among: scope, time, cost and quality; stakeholders with differing needs and expectations; identified requirements (needs) and unidentified requirements (expectations).

An important task in the project management is the accomplishment of a plan for the project execution.

A project's example is presented in figure 1. This plan represents a construction of radio base station. A radio base station is a product developed for telecommunications companies, and has as objective to make possible communication between devices of a mobile telephone system.

The example presented in figure 1 is a simplified case that represents a set of tasks and their dependence relations.

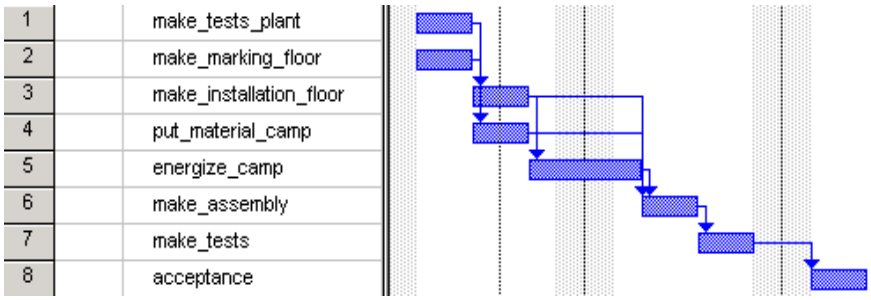

Fig. 1. Gantt Chart example of the radio station base construct

3 Situation Calculus

Shanahan [9], presents Situation Calculus as being a logical formalism whose elements are actions, effects and situations:

- *situations*: are full description of a specific situation of the real world;
- *fluents*: are functions that denote properties or relations that can change a situation to another one;
- *actions*: transforms a situation into another one;
- $Result(\alpha, s)$: α denotes the resulting situation after executing the action α in situation s;
- $Holds(\delta, s)$: this predicate defines that this fluent δ is valid in the situation s[1].

Besides these elements, the Situation Calculus has three classes of formulae: effect axioms, domain constraints and observation sentences.

An *effect axiom* is a formula of the form $\forall s[Holds(\delta, Result(\alpha, s))]$ or $\forall s[Holds(\delta, Result(\alpha, s)) \leftarrow Holds(\delta_i, s)]$ and has the purpose of describing action effects.

A *domain constraint* is a formula of the form $\forall s[Holds(\delta, s) \leftarrow Holds(\delta_i, s)]$ and its objective is describe eventual domain constraints.

A collection of effect axioms and domain constraints is called *domain description*. In domain description must exist an effect axiom for each action.

An *observation sentence* is a formula in which every occurrence of $Holds$ is of the form $Holds(\delta, \sigma)$ or $\neg Holds(\delta, \sigma)$, where σ is a non-variable situation term, for example: $Holds(ready\_begin, s0)$.

The Situation Calculus cannot represent time, concurrent actions and actions with duration [9, 1, 2].

Hence, to be able to use Situation Calculus as formalism to represent actions and changes in a project management domain, it is necessary to develop a representation extension.

[1] It is assumed that the goal is to represent the fact that is raining in the situation. There is a way to do this, using notation $Raining(S0)$. Another alternative is to express the fact as $Holds(Raining, s0)$.

4 Associating a situation with time

Shanahan shows in [9], that there are two ways to interpret the idea of a situation in the Situation Calculus, both compatible with the original proposal of *McCarthy and Hayes*, who define a situation as *"the complete state of the universe at an instant of time"* [7].

The first way is think of a situation as being defined by the set of fluents that holds in it. The second way is think of situations defined by the *Result* function. According to this interpretation, there are the following axioms, which *Shanahan* call the *axioms of arboreality*:

$$Result(\alpha_1, s_1) = Result(\alpha_2, s_2) \rightarrow \alpha_1 = \alpha_2 \wedge s_1 = s_1 \tag{1}$$

$$S0 \neq Result(\alpha, s) \tag{2}$$

In this work, we chose the second way, that defines a situation as an unique node in a tree of situations defined by the *Result* function.

The following restrictions will be assumed: (1) the domain had only discrete units of time, and; (2) the domain has only instantaneous actions.

The relation of the time point with each node in a tree of situations, can be represented in this form:

$$Time(S0) = 0 \tag{3}$$

where 0 is an arbitrarily assigned value, and

$$Time(Result(a, s)) = Time(s) + 1 \tag{4}$$

Hence, each situation of the tree of situations will be associated with one time point (see figure 2).

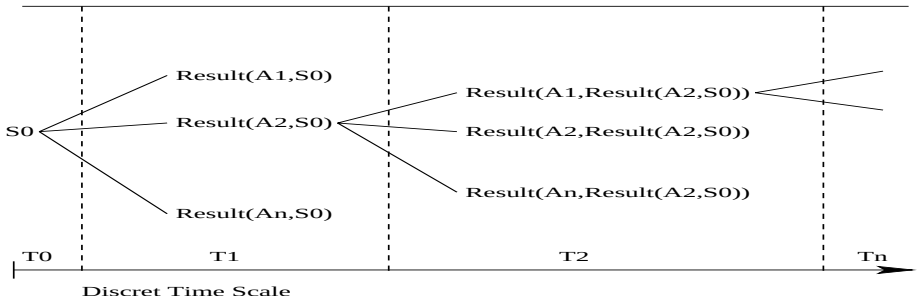

Fig. 2. The tree of Situations associanting with time point

5 Representing actions that consumes resources

The word *resources* in project management refers to objects that can be consumed, or "borrowed" during the execution of plans, constraining the possible actions[8, 4, 6]. E.g., driving a car requires and cosumes fuel, building a house requires and consume bricks, etc.

To represent resources in the domain, it is necessary to give the language the ability to express a pre-condition of the type $have(\$2, 50)$ and the planner can execute this in an efficient way. .

Resources normally are represented as variable that can assume real or integer values [8, 4, 6]. In this paper, all resources are represented through monetary values.

The formula represents the amount of resources that an action α needs in order to be executed:

$$\forall\alpha, \exists\xi[Resource(\alpha, \xi)] \tag{5}$$

where $\xi \geq 0$ and $\xi \in \mathbb{R}^2$.

This new formula is added to the effect axiom:

$$\forall s[Holds(Total, \delta, Result(\alpha, s)) \leftarrow Holds(\beta, \delta_i, s) \wedge Resource(\alpha, \xi)] \tag{6}$$

Where $Total$ it is a variable that represents on the resources that is being expended for the execution in the situation s. The result of $Total$ is equal to β plus ξ.

Formula (6) says that the action α to be executed *needs* resource ξ and during the execution *consumes* the resource ξ.

Besides modifying the effect axiom's form, it is necessary to modify the observation sentence's form:

$$Holds(\beta, \delta, S0) \tag{7}$$

where β is the resources value expenses in the initial situation. The value of resources expenses in the initial situation is always zero, because no action were executed yet.

6 Representing resources expenses over time

One of the objectives of this work is construct an architecture capable of representing the amount of resources expenses in a specific instant of time and/or how much it will be necessary to spend to execute a given task. For example, given a plan, what amount of resources is necessary in time point τ ?

It is possible to represent resources that each action spends. It is also possible to associate each situation in one instant of time. But, there is not a direct

[2] $\mathbb{R}$ means the real numbers set

mapping between resources and time. The representation of resources expenses over time is defined by the following formulas:

$$ResourceTime(S0, Time(S0), 0) \tag{8}$$

$$ResourceTime(Result(\alpha, s), \tau, \xi) \leftarrow Time(Result(\alpha, s), \tau) \wedge Resource(\alpha, \xi) \tag{9}$$

$$ResourceTime(Result(\alpha, s), \tau, \xi) \leftarrow \neg Time(Result(\alpha, s), \tau) \wedge ResourceTime(s, \tau, \xi) \tag{10}$$

7 A Planner Based on the Extend Situation Calculus

The Situation Calculus axioms can be used to construct logical programs to plan or to supply prognostics.

Shanahan [9] describes a program of Situation Calculus as a conjunction of the *universal frame axiom*[3], in the form $Holds(\delta, result(\alpha, s)) \leftarrow Holds(\delta, s) \wedge \neg Affects(\alpha, \delta, s)$, where the predicate $Affects(\alpha, \delta, s)$ defined that fluente δ is affected by the action α in situation s; a finite set of *observation sentences*, with the format: $Holds(\delta, s0)$; a finite set of *effect axioms*, in the form: $Holds(\delta, result(\alpha, s)) \leftarrow Holds(\delta_i, s)$; and a finite set of clauses $Affects$.

Hence, a *Situation Calculus program that implements actions that consume resources and represent time is the conjuntion of*:

- the universal frame axiom:

$$Holds(Total, \delta, Result(\alpha, s)) \leftarrow Holds(\beta, \delta, s) \wedge notAffects(\alpha, \delta, s) \wedge Resource(\alpha, \xi) \tag{11}$$

  where $\beta + \xi = Total$.
- a finite set of $Affects$ clauses of the form:

$$Affects(\alpha, \delta, s) \tag{12}$$

- a finite set of observation sentences of the form:

$$Holds(\beta, \delta, S0) \tag{13}$$

  where β is the resources expenses value in the initial situation, and is always equal zero.

[3] When the Situation Calculus is used to describe effects of actions, besides describing changes it is necessary to describe what remains unchanged (the frame problem). In order to describe what remains unchanged, the Situation Calculus has the universal frame axiom [9].

– a finite set of effect axioms of the form:

$$Holds(Total, \delta, Result(\alpha, s)) \leftarrow \Pi \wedge Resource(\alpha, \xi) \tag{14}$$

where Π does not mention the $Affects$ predicate and every occurrence of the $Holds$ predicate in Π is of the form $Holds(\beta, \delta_i, s)$, where $\beta + \xi = Total$.

– a finite set of *resources observation sentences* of the form:

$$\forall\alpha[Resource(\alpha, \xi)] \tag{15}$$

– the time axioms of the form:

$$Time(S0) = 0 \tag{16}$$

where 0 is an arbitrarily assigned value, and

$$Time(Result(\alpha, s)) = Time(s) + 1 \tag{17}$$

– the *resources expenses by time* axioms of the form:

$$ResourceTime(S0, Time(S0), 0) \tag{18}$$

$$ResourceTime(Result(\alpha, s), \tau, \xi) \leftarrow Time(Result(\alpha, s), \tau) \wedge Resource(\alpha, \xi) \tag{19}$$

$$ResourceTime(Result(\alpha, s), \tau, \xi) \leftarrow \neg Time(Result(\alpha, s), \tau) \wedge ResourceTime(s, \tau, \xi) \tag{20}$$

– a finite set of sentences of general clauses (*background sentences*) not mention the predicates $Holds$ or $Affects$.

These formulas can easily be transformed into clauses of a logical program. For example, there is a clause for formula (13) that can be shown in the figure 3. The term *ready_begin* is a fluent valid in the initial situation $s0$.

```
1 holds(0,ready_begin,s0).
```

Fig. 3. Observation sentence code (initial state).

Another example is the effect axioms code (14) which can be shown in the figure 4. The term *make_installation_floor* is an action that can executed in situation S, if the fluents *make_tests_plant* and *installation_floor* are valid. This action spends resources total which is the sum of resources spent to make

the fluents *test_plant* and *marking_floor* to be true (Y) plus the resources spent to execute the action *make_installation_floor* (X).

```
holds(Total,installation_floor,result(make_installation_floor,S)):-
        holds(Y,make_tests_plant,S),
        holds(Y,make_marking_floor,S),
        resource(make_installation_floor,X),
        Total is Y + X.
```

Fig. 4. Effect axiom code.

The time axioms (16) and (17) are codified as described in the figure 5.

```
time(s0,0).

time(result(A,S),T):-
        time(S,T1),
        T is T1 + 1.
```

Fig. 5. Time axioms code.

The axioms for representating of resources expenses over one instant of time are transformed as described in the figure 6.

```
resourceTime(s0,0,0).

resourceTime(result(A,S),T,Res):-
        time(result(A,S),T),
        resource(A,Res).

resourceTime(result(A,S),T,Res):-
        \+ time(result(A,S),T),
        resourceTime(S,T,Res).
```

Fig. 6. Resources expenses over time axioms code.

The universal frame axiom (12) is transformed as described in the figure 7.

```
holds(Total,F,result(A,S)):-
        holds(Y,F,S),
        not(affects(A,F,S)),
        resource(A,X),
        Total is Y + X.
```

Fig. 7. Universal frame axiom code.

Where the predicate *not* is an implementation of SLDNF-resolution [5].

8 Extendented Situation Calculus Implementation in domain of Project Management

This section will demonstrate the representation of the actions applications in a simple radio base station construction project, as shown in the figure 1.

Information about project actions that must be represented are: the precedence between the actions, or either, the pre-conditions of each action; the description of the effect that the actions can generate, either positive[4] or negative[5]; and the resource (expressed in monetary units) that each action needs. These information can be visualized in the table 1.

Table 1. Domain actions description

Action	**Pre-condition**	**Effects**	**Resources**
make_tests_plant	ready_begin	test_plant	10 MU
make_marking_floor	ready_begin	marking_floor	15 MU
make_installation_floor	test_plant marking_floor	installation_floor ¬ marking_floor	10 MU
put_material_camp	test_plant	material_camp	20 MU
make_energize_camp	installation_floor	energize_camp	30 MU
make_assembly	material_camp energize_camp installation_floor	mounted	20 MU
make_tests	mounted	tests	15 MU
acceptance	tests	radio_station	10 MU

For modelling these information in logic programming, we need to write in accordance with the axioms presented in the last section.

After writing clauses that represents these information, we obtain a knowledge base wich represents the domain's capability.

[4] Effects that modify the fluent value for true.
[5] Effects that modify the fluent value for false.

Applying a deductive mechanism on this knowledge base we can generate plans or estimates, depending on the goal clause given. For example, if the goal clause is $holds(Total, w, result(make\_tests\_\ plant, s0))$, the deductive mechanism applied on the knowledge base return one estimate of the resultant situation. The search tree is showing in the figure 8.

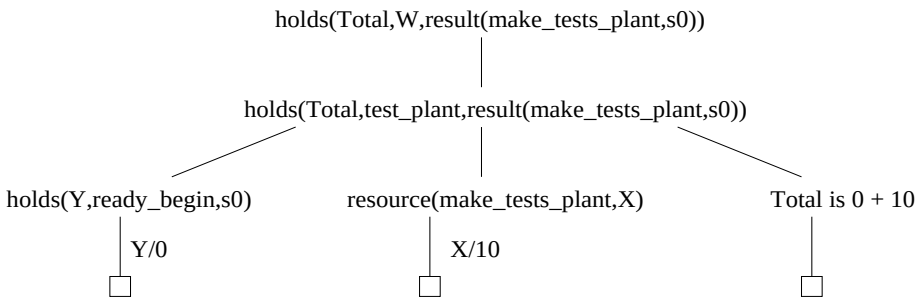

Fig. 8. Search tree

On the other hand, if the goal clause is, for example, $holds(Total, energize\_camp, X)$, the deductive mechanism applied on the knowledge base generate one or more plans that becomes fluent *energize_camp* true. This plan returned in the format $result(\alpha_1, \cdots, result(\alpha_m, s0))$ unified with the variable X. And the variable $Total$ returns the total value from resources expenses for the plan.

The goal clauses can be expressed in other forms, restricting the plan, as shown by following formulas, figure 9.

```
1  :- holds(Total,radio_station,Plan), Time(Plan,T), T =< 10.
2  :- holds(Total,radio_station,Plan), resourceTime(Plan,2,X), X =< 10.
```

Fig. 9. Object clauses code.

The goal in the first line asks for a plan that return *radio_station* in the maximum 10 units of time. While the goal in the second line asks for a plan that return *radio_station* and the maximum 10 MU of resources spend in time t2.

9 Conclusion and future work

This work presented an extension of Situation Calculus that makes possible representation of actions that consumes resources and associations of each situation with one time point.

The analysis of presented examples makes clear that extend Situation Calculus can be used in domains which need the representation of actions that consume resources and need representation of time.

To achieve full representation requirements for project management domain planners, it is necessary to develop an additional extension of Situation Calculus, in order to make possible the representation of concurrent actions and actions with duration.

Future work includes the extension of current calculus to represent concurrent actions, actions with duration, time and resources at the same time.

References

1. José Júlio Alferes, Renwei Li, and Luís Moniz Pereira. Concurrent actions and changes in the situation calculus. *Proc. of IBERAMIA*, pages 93–104, 1994.
2. Jorge Baier and Javier Pinto. Non-instantaneous actions and concurrency in the situation calculus. *In 10th European Summer School in Logic, Language and Information*, 1998.
3. Project Management Institute Standards Committee. *A Guide to the Project Management Body of Knowledge*. Automated Graphic Systems, Maryland, USA, 1996.
4. K. Currie and A. Tate. O-plan: the open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991 1991.
5. Kees. Doets. *From Logic to Logic Programming*. Massachusetts Institute of Technology, 1994.
6. A. El-Kholy and B. Richards. Temporal and resource reasoning in planning: the parcplan approach. *Proc. ECAI-96*, pages 614–618, 1996.
7. J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.
8. Stuart J. Russel and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, 1995.
9. Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. The MIT Press, 1997.