Planning with failure

Fausto Giunchiglia[1,2], Luca Spalazzi[1,3], Paolo Traverso[1]
IRST, 38050 Povo, Trento, Italy
University of Trento, Via Inama 5, 38100 Trento, Italy
University of Ancona, Via Brecce Bianche, 60131 Ancona, Italy
{fausto,spalazzi,leaf}@irst.it

Abstract

Many real world applications require systems with both reasoning and sensing/acting capabilities. However, most often, these systems do not work properly, i.e. they fail to execute actions and rarely perceive the external world correctly. No action, even if apparently simple, is guaranteed to succeed and, therefore, no planning can be "sound" (with respect to the real world) without taking into account *failure*. In this paper, we present a theory of planning that provides (1) a language that allows us to express failure; (2) a declarative formal semantics for this language; (3) a logic for reasoning about (possibly failing) plans.

Failure

Many real world applications require systems with both reasoning and sensing/acting capabilities. Reasoning allows systems to achieve high level goals. Acting and sensing allows them to work in a complex and unpredictable external environment. Most often, systems sensing and acting in the world do not work properly, i.e. they fail to execute actions and rarely perceive the external world correctly. This is mainly due to the intrinsic complexity of reality and to the fact that actuators and sensors are not perfect (e.g. in a navigating robot, sonars are not precise enough). As a consequence, no action, even if apparently simple, is guaranteed to succeed and, therefore, no planning can be "sound" (with respect to the real world) without taking into account *failure*.

In this paper, we propose a theory with the following main characteristics:

1. The basic operations for dealing with failure are represented explicitly within plans. These operations serve as the basic constructs for a variety of control mechanisms that can be used to detect and recover from failure.
2. The planning language has a declarative formal semantics. This semantics formalizes failure and success in plan execution, failure detection and reaction to failure.
3. The planning language is embedded in a logic. This logic provides the ability to reason deductively about failure.

We use the proposed theory for reasoning about plans. This gets us close to the work on classical planning. The fact that we perform reasoning inside a logical theory gets us close to the work on deductive planning. Nevertheless, the work described in this paper is different in spirit and in methodology from most of the research on classical and deductive planning. A fundamental assumption underlying classical (deductive) planning is that it is possible to state conditions (axioms) sufficient to guarantee successful action executions. Based on this assumption, the research on classical (deductive) planning has focused on the problem of plan generation, i.e. on providing powerful and efficient heuristics (deduction strategies) to build "good" plans, i.e. plans that are guaranteed to succeed (or more likely, that have good chances to succeed). However, in most real world applications, there is in general no way to build a priori "good" plans. Most often, the problem is not only to generate good plans, but also to reason about execution control and monitoring. The emphasis of this paper is therefore on providing a theory which allows us to represent and reason about actions and plans that are not guaranteed to succeed. The natural development of this work is a theory that includes, but is not limited to, plan generation, i.e. a theory for reasoning about all the activities of a planning agent, e.g. execution control and monitoring.

The intuition

A theory that takes into account failure has to deal with some fundamental novel issues. Let us explain this through a simple example. Given a sequence of actions α and β, usually written as $\alpha;\beta$, consider the problem of providing semantics to the *execution of the sequence* $\alpha;\beta$. Most of the classical planners execute $\alpha;\beta$ in the following way: α is executed first; if α suc-

ceeds, they execute β afterwards; if α fails, they do not execute β and recover from failure, e.g. by replanning. Consider now, as an example, the Dynamic Logic semantics of $\alpha;\beta$, i.e. $\rho(\alpha;\beta)$, where ρ is the relation describing accessibility between pairs of states (Harel 1984). $\rho(\alpha;\beta)$ is defined as $\rho(\alpha)\cdot\rho(\beta)$, i.e. the concatenation of $\rho(\alpha)$ and $\rho(\beta)$, when α succeeds as well as when α fails. But when α fails, α, and only α, not β, gets executed and changes the world. Therefore, $\rho(\alpha;\beta)$ does not take into account failure.

The idea is to provide a semantics that takes into account failure. For instance, let $then(\alpha,\beta)$ denote the execution of a sequence of actions (we use the symbol *then* to distinguish it from the usual symbol for sequences, i.e. ;). Intuitively, we build a semantics where $S(\alpha)$ $[F(\alpha)]$ denotes the fact that α succeeds [fails]. We have that

$$S(then(\alpha,\beta)) = S(\alpha)\cdot S(\beta) \quad (1)$$

$$F(then(\alpha,\beta)) = F(\alpha)\cup S(\alpha)\cdot F(\beta) \quad (2)$$

(1) states that the execution of the sequence succeeds ($S(then(\alpha,\beta))$) iff both the actions succeed ($S(\alpha)\cdot S(\beta)$). (2) states that the execution of the sequence fails iff either the first action fails ($F(\alpha)$) and the second is not executed or the first succeeds and the second fails ($S(\alpha)\cdot F(\beta)$). This topic is described in detail in Section "A planning language with failure".

Consider now the Dynamic Logic axiom used to reason about sequences, i.e. $[\alpha;\beta]p \leftrightarrow [\alpha]([\beta]p)$, where p is a wff representing the properties holding after the execution of $\alpha;\beta$. This axiom does not distinguish explicitly the case where α fails. We build a logical theory where the wff $T(\alpha,\text{"}S\text{"})$ $[T(\alpha,\text{"}F\text{"})]$ means intuitively that α succeeds [fails]. In general, $T(\alpha,\text{"}p\text{"})$ stands for "the proposition p holds after executing α". This theory allows for reasoning about failure, i.e. for reasoning about the state of the world after that failure occurs. For instance, we can prove the following theorem

$$T(\alpha,\text{"}F\text{"})\wedge T(\alpha,\text{"}p\text{"})\supset T(then(\alpha,\beta),\text{"}p\text{"}) \quad (3)$$

which states that, if p holds after that α fails, then p holds after executing the sequence $then(\alpha;\beta)$. This topic is described in detail in Section "Reasoning about failure".

A planning language with failure

The syntax

In MRG, plans are expressions called *tactics*. The syntax of tactics is based upon a set of symbols $\mathcal{T}_0$, that we call *primitive tactics*[1]. Intuitively, this set represents basic actions. For instance, $goto(A)$ can be a basic (even if very complex) action which moves a robot in a location (position) A, e.g. by means of an in-door navigation system. From $\mathcal{T}_0$, we inductively construct the set of tactics $\mathcal{T}$. For lack of space we do not describe the full syntax of MRG. We focus on the subset of the language that deals with failure.

1. $\Sigma\in\mathcal{T}$, $\Phi\in\mathcal{T}$, $\mathcal{T}_0\subseteq\mathcal{T}$
2. if $\alpha,\beta,\gamma\in\mathcal{T}$, then **iffail** α **then** β **else** $\gamma\in\mathcal{T}$,

Σ and Φ represent the basic actions that generate success and failure, respectively. Σ [Φ] does nothing else but terminate execution in a state of success [failure]. The intended meaning of **iffail** α **then** β **else** γ is: "do α, if α fails do β, otherwise do γ". The full syntax of MRG allows for tactics that include conditional expressions of the form *if* P *then* α *else* β, loops of the form *while* P *do* α and recursive definitions. Moreover, it allows us to represent explicitly the internal planning activities of the planner itself, like plan generation, plan execution and information acquisition (see (Traverso, Cimatti, & Spalazzi 1992)). Finally, it allows expressions representing goals to be achieved (e.g. $At(A)$, which corresponds to the goal satisfied when the robot is at the location A in a building). The full syntax of the MRG planning language is given in (Traverso & Spalazzi 1993)

A remark. The tactics Σ and Φ allow the planner to state explicitly whether a tactic fails or succeeds. Success and failure of a tactic does not coincide necessarily with the achievement/not-achievement of a related goal. The former is a property of tactics, the latter is a relation between tactics and goals. A tactic may fail and, nevertheless, achieve the goal the tactic has been executed for. For instance, the tactic $goto(A)$ that moves a robot to a given location A, can be executed to achieve the goal $At(A')$, i.e. the world state where the robot is in an area A' that contains A. The tactic, when executed, may fail when the robot is in A' but not in A. It therefore achieves the goal. Vice versa, a tactic may succeed and may not achieve the goal, e.g. in the case the tactic $goto(A')$ is executed to achieve the goal $At(A)$. This is a possible situation in systems where the actuators and sensors may not allow for executing tactics precise enough to guarantee the achievement of certain goals, e.g. a precise robot position. Other tactics can thereafter be executed to try to decide whether the goal has been achieved.

We extend the language with the following definitions which express some of the constructs that can be used to control failure:

1. $then(\alpha,\beta)$ = **iffail** α **then** Φ **else** β,

[1] More precisely, primitive tactics are constructed from a set of usual first order terms and a set of *tactic symbols*, that, intuitively represent action types. For lack of space we skip the formal definition of the syntax of primitive tactics.

2. $orelse(\alpha, \beta) =$ **iffail** α **then** β **else** Σ,
3. $;(\alpha, \beta) =$ **iffail** α **then** β **else** β,
4. $repeat(\alpha) = orelse(then(\alpha, repeat(\alpha)), \Sigma)$

then captures the intended meaning of the execution of a sequence of actions that takes into account failure. If the first action fails, it does not execute the second, but simply terminates execution with failure. *orelse* captures and reacts to failure: $orelse(\alpha, \beta)$ reacts to failure of α by executing β. Since β can be an internal planning activity, e.g. plan generation or plan execution, *orelse* can represent recovery from failure by replanning (like in most classical planners, e.g (Wilkins 1985)) as well as recovery from failure by executing special purpose exception handling routines (like in most embedded planners, e.g. (Georgeff & Lansky 1986; Simmons 1990)). The construct ; resembles the sequential composition of Dynamic Logic. The second action is executed anyway, independently of the failure/success of the first action. *repeat* is recursively defined. It repeats the execution of α till α fails. If α never fails, execution does not terminate. For instance, $repeat(\Sigma)$ executes forever. Notice that if it terminates, $repeat(\alpha)$ always succeeds, since *orelse* in the definition of *repeat* reacts to failure by executing Σ, which always succeeds. For instance, $repeat(\Phi)$ is equivalent to $orelse(\Phi, \Sigma)$ which, after Φ, executes Σ and succeeds.

As a simple example, consider a robot that tries to move a block from the table. The robot does not manage to move the block since it is steadily attached to the table. Consider the problem of expressing this apparent failure with a success, since the robot is not trying to move the block on top of another block, but is simply testing whether the block is safely attached to the table. In MRG this is captured by the following tactic:

iffail $move(block)$ **then** Σ **else** Φ

The construct **iffail**, together with the basic actions for success and failure, allows us to express the fact that if the action $move(block)$ succeeds, then we have failure and vice versa.

The semantics

The semantics of MRG is defined relative to a given structure $\mathcal{U}$, of the form [2]

$$\mathcal{U} = (\mathcal{W}, \mathcal{B}, \mathcal{S}, \mathcal{F})$$

where $\mathcal{W}$ is the usual set of states; $\mathcal{B}$ is a set, the elements of which are called *behaviours*, i.e. sequences of states. A tactic α is interpreted into a set of behaviours. This set is divided into two disjoint subsets: the set $\mathcal{S}(\alpha)$ of successful behaviours, that we call the *success set* of α, and the set $\mathcal{F}(\alpha)$ of behaviours that fail, that we call the *failure set* of α. $\mathcal{S}$ and $\mathcal{F}$ map a primitive tactic into a set of behaviours. Formally:

$$\mathcal{S} : \mathcal{T}_0 \rightarrow 2^{\mathcal{B}} \qquad \text{and} \qquad \mathcal{F} : \mathcal{T}_0 \rightarrow 2^{\mathcal{B}}$$

The success and failure set of Σ and Φ are given below.

$$\mathcal{S}(\Sigma) = \mathcal{W} \qquad \mathcal{F}(\Sigma) = \emptyset \tag{4}$$
$$\mathcal{S}(\Phi) = \emptyset \qquad \mathcal{F}(\Phi) = \mathcal{W} \tag{5}$$

Σ always succeeds. Formally, this is captured by the fact that its failure set is empty. Σ does nothing else but succeeding. Formally, this means that its success set is the set of behaviours composed by any single state, i.e. $\mathcal{W}$. For instance, we have that, for any tactic α, $\mathcal{S}(\alpha) \cdot \mathcal{S}(\Sigma) = \mathcal{S}(\alpha)$. Vice versa, the success set of Φ is empty while its failure set is $\mathcal{W}$.

The mappings $\mathcal{S}$ and $\mathcal{F}$ are extended inductively to supply meanings for the full set of tactics $\mathcal{T}$. In the following we extend the definition of $\mathcal{S}$ and $\mathcal{F}$ for the subset of the language of MRG defined in this paper, i.e. for the construct **iffail**. We write $b_1 \cdot b_2$, where $b_1, b_2 \in \mathcal{B}$, as the concatenation of the two behaviours b_1 and b_2. This operation is extended to sets of behaviours in the usual way.

$$\begin{aligned} &\mathcal{S}(\textbf{iffail } \alpha \textbf{ then } \beta \textbf{ else } \gamma) = \\ &\qquad \mathcal{S}(\alpha) \cdot \mathcal{S}(\gamma) \cup \mathcal{F}(\alpha) \cdot \mathcal{S}(\beta) \\ &\mathcal{F}(\textbf{iffail } \alpha \textbf{ then } \beta \textbf{ else } \gamma) = \\ &\qquad \mathcal{S}(\alpha) \cdot \mathcal{F}(\gamma) \cup \mathcal{F}(\alpha) \cdot \mathcal{F}(\beta) \end{aligned} \tag{6}$$

In the following we show the semantics of *then* and *orelse*, which is derived according to their definitions in terms of **iffail**, Σ and Φ.

$$\begin{aligned} &\mathcal{S}(then(\alpha, \beta)) = \mathcal{S}(\alpha) \cdot \mathcal{S}(\beta) \\ &\mathcal{F}(then(\alpha, \beta)) = \mathcal{F}(\alpha) \cup \mathcal{S}(\alpha) \cdot \mathcal{F}(\beta) \\ &\mathcal{S}(orelse(\alpha, \beta)) = \mathcal{S}(\alpha) \cup \mathcal{F}(\alpha) \cdot \mathcal{S}(\beta) \\ &\mathcal{F}(orelse(\alpha, \beta)) = \mathcal{F}(\alpha) \cdot \mathcal{F}(\beta) \end{aligned} \tag{7}$$

repeat, which is recursively defined, is interpreted by building a fix point.

Reasoning about failure

MT^* is a first order classical metatheory which provides a deduction machinery for reasoning about MRG tactics (and thus about failure). MT^* has the following characteristics:

[2] Actually, $\mathcal{U}$ contains also a domain set $\mathcal{D}$, but this is not relevant here.

1. The language of MRG is "embedded" in MT^*. MRG tactics become terms of MT^*. Intuitively, they denote the state of affairs after that a tactic has been executed.
2. MT^* allows wffs of the kind $T(\alpha, \text{“}p\text{”})$, where T is a predicate symbol, α is a tactic and “p” is a constant denoting the wff p. Intuitively, the intended meaning of $T(\alpha, \text{“}p\text{”})$ is “p holds after executing α”.
3. MT^* represents failure explicitly by means of a proposition F. $T(\alpha, \text{“}F\text{”})$ $[T(\alpha, \text{“}\neg F\text{”})]$ intuitively means that the execution of α fails [succeeds].

Success is defined as the proposition S, where $S \leftrightarrow \neg F$. Tactics in MT^* are ground terms. The set of tactics in MT^* is constructed in the same way as the set $\mathcal{T}$ in MRG, where: $\mathcal{T}_0$ is a set of constants; Σ and Φ are constants; iffail is a 3-place function symbol; *then*, *orelse*, ; and *repeat* are defined function symbols (e.g. $\forall\alpha\forall\beta\ then(\alpha,\beta) = \text{iffail } \alpha \text{ then } \Phi \text{ else } \beta$). In the following, we give some of the axioms of MT^*. We sometimes write α, β and γ either as variables of sort tactic or as tactics (i.e. ground terms). The distinction should be clear from the context.

The following axioms state that after executing an action, we have either failure or success. For any tactic α we have:

$$\begin{array}{ll}(A1) & T(\alpha, \text{“}S\text{”}) \vee T(\alpha, \text{“}F\text{”}) \\ (A2) & \neg(T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}F\text{”}))\end{array} \quad (8)$$

The following axioms state the properties of Σ and Φ. For any proposition p, constant “p” of MT^*, we have:

$$\begin{array}{ll}(A3) & T(\Sigma, \text{“}S\text{”}) \\ (A4) & T(\Phi, \text{“}F\text{”}) \\ (A5) & \text{“}p\text{”} \neq \text{“}F\text{”} \wedge \text{“}p\text{”} \neq \text{“}S\text{”} \supset (T(\Sigma, \text{“}p\text{”}) \leftrightarrow p) \\ (A6) & \text{“}p\text{”} \neq \text{“}F\text{”} \wedge \text{“}p\text{”} \neq \text{“}S\text{”} \supset (T(\Phi, \text{“}p\text{”}) \leftrightarrow p)\end{array} \quad (9)$$

Axioms (A3) and (A4) state that Σ always succeeds and Φ always fails. Axioms (A5) and (A6) state that Σ and Φ do not change the world but the fact that they force success or failure.

The following axiom states the properties of iffail. For any tactic α, β and γ, for any constant “p”, we have:

$$\begin{array}{ll}(A7) & T(\text{iffail } \alpha \text{ then } \beta \text{ else } \gamma, \text{“}p\text{”}) \leftrightarrow \\ & ((T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}T(\gamma, \text{“}p\text{”})\text{”})) \vee \\ & (T(\alpha, \text{“}F\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}p\text{”})\text{”})))\end{array} \quad (10)$$

Axiom (A7) states that p holds after executing iffail iff it holds after executing α with success and γ afterwards, or α with failure and β afterwards.

From the axioms above and the definitions of *then* and *orelse* we can prove the following theorems:

$$\begin{array}{ll}(T1) & T(then(\alpha,\beta), \text{“}p\text{”}) \leftrightarrow \\ & ((T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}p\text{”})\text{”})) \vee \\ & (T(\alpha, \text{“}F\text{”}) \wedge T(\alpha, \text{“}p\text{”}))) \\ (T2) & T(orelse(\alpha,\beta), \text{“}p\text{”}) \leftrightarrow \\ & ((T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}p\text{”})) \vee \\ & (T(\alpha, \text{“}F\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}p\text{”})\text{”})))\end{array} \quad (11)$$

Theorem (T1) can be used to reason about the execution of sequences of actions that may fail. In the right hand of the equivalence we have two cases. If α succeeds ($T(\alpha, \text{“}S\text{”})$), then p holds after the execution of β ($T(\alpha, \text{“}T(\beta, \text{“}p\text{”})\text{”})$). If α fails ($T(\alpha, \text{“}F\text{”})$), p holds after the execution of α ($T(\alpha, \text{“}p\text{”})$). Theorem (T2) can be used to reason about failure capturing and recovery from failure.

From theorems (T1) and (T2) we can easily prove the following theorems. Notice the relation between these theorems and the success and failure sets of *then* and *orelse* reported in (7).

$$\begin{array}{ll}(T3) & T(then(\alpha,\beta), \text{“}S\text{”}) \leftrightarrow \\ & (T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}S\text{”})\text{”})) \\ (T4) & T(then(\alpha,\beta), \text{“}F\text{”}) \leftrightarrow \\ & (T(\alpha, \text{“}F\text{”}) \vee (T(\alpha, \text{“}S\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}F\text{”})\text{”}))) \\ (T5) & T(orelse(\alpha,\beta), \text{“}S\text{”}) \leftrightarrow \\ & (T(\alpha, \text{“}S\text{”}) \vee (T(\alpha, \text{“}F\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}S\text{”})\text{”}))) \\ (T6) & T(orelse(\alpha,\beta), \text{“}F\text{”}) \leftrightarrow \\ & (T(\alpha, \text{“}F\text{”}) \wedge T(\alpha, \text{“}T(\beta, \text{“}F\text{”})\text{”}))\end{array} \quad (12)$$

Some remarks. In classical planning, the behaviour of basic actions can be described by axioms of the following form:

$$p_1 \wedge \ldots \wedge p_n \supset T(\alpha, \text{“}q_1 \wedge \ldots \wedge q_m\text{”})$$

These axioms state that, if the preconditions $p_1, \ldots, p_n$ hold, then the effects of the execution of α are $q_1, \ldots, q_m$. But the actual effects of α depend on an infinite number of preconditions, that no model of the world can capture. The actual effects depend on whether α will succeed or fail. In MT^*, the fact that failure is explicit can be used to give a more "realistic" axiomatization that states that *if α will not fail*, then some facts will hold after its execution, e.g.:

$$T(\alpha, \text{“}\neg F\text{”}) \wedge p_1 \wedge \ldots \wedge p_n \supset T(\alpha, \text{“}q_1 \wedge \ldots \wedge q_m\text{”}) \quad (13)$$

We can define failure (and similarly success) for a primitive tactic α with axioms of the form

$$T(\alpha, \text{“}F\text{”}) \leftrightarrow T(\alpha, \text{“}r_1\text{”}) \wedge \ldots \wedge T(\alpha, \text{“}r_k\text{”}) \qquad (14)$$

i.e. the planner decides that an action fails when the conditions $r_1, \ldots, r_k$ are observed from the world after that α has been executed. Notice the difference between axioms (14) and axioms of the form

$$r_1 \wedge \ldots \wedge r_k \supset T(\alpha, \text{“}F\text{”}) \qquad (15)$$

In (15), any r_i is a condition over the world before that α gets executed. In (14), any $T(\alpha, \text{“}r_i\text{”})$ is a condition after the execution of α. In (15), failure is predicted. Axioms (14) can instead be used when failure cannot be predicted. As a simple example, we can define failure and success of navigation in an environment where it is impossible to predict the presence of unavoidable obstacles as follows:

$$T(goto(A), \text{“}\neg F\text{”}) \leftrightarrow T(goto(A), \text{“}At(A)\text{”})$$

$$T(goto(A), \text{“}F\text{”}) \leftrightarrow T(goto(A), \text{“}\neg At(A)\text{”})$$

where $At(A)$ means that the robot is at A.

Finally, in MT^* we can *assume that* an action succeeds (or fails) and deduce consequences depending on that assumption. That is, we can hypothetically reason about failure without knowing whether the action will actually fail or succeed. For instance, we can assume $T(\alpha, \text{“}\neg F\text{”})$ and derive all the possible interesting consequences that hold whenever α succeeds, e.g. $T(\alpha, \text{“}p\text{”})$. We can assume that either α does not fail or failure of α is recovered by β afterwards. i.e. $T(orelse(\alpha, \beta), \text{“}\neg F\text{”})$, and derive that, under that hypothesis,

$$T(\alpha, \text{“}p\text{”}) \vee T(\alpha, \text{“}T(\beta, \text{“}q\text{”})\text{”})$$

where q is a property that holds after that α fails and β succeeds.

Related work

As far as we know, the approach presented in this paper has never been proposed before. So far, failure has had very little attention in formal theories of actions. This is the case of Dynamic Logic (Harel 1984), as we discussed in Section "The intuition", and, as a consequence, of the approaches to planning based on Dynamic Logic, e.g. (Rosenschein 1981). This is also the case of the work described in (Steel 1993), which shares with our research most of the underlying motivations. The idea that actions in the real world may fail is, of course, present in most of the work on reasoning about actions and plans. But none of them provides a theory where failure is represented explicitly in the language, in the semantics and in the logic. The usual assumption is that it is possible to assert axioms that define the conditions sufficient to guarantee successful action attempts (see for instance (Allen & Ferguson ; Lifschitz 1993)). This assumption is not realistic for the real world applications we are interested in.

A lot of recent research in planning is more and more focusing on real world applications where failure is a key aspect. This is the case of the research on "embedded planners". Several embedded planners have been proposed so far (see for instance (Georgeff & Lansky 1986; Simmons 1990; Beetz & McDermott 1992; Firby 1992; Spalazzi, Cimatti, & Traverso 1992b)) and have been successfully applied in particular application domains (like mobile robots (Georgeff, Lansky, & Schoppers 1986; Simmons 1991; Cimatti *et al.* 1992) and fault diagnosis for real time systems (Georgeff & Lansky 1986)). Most of them provide different and flexible failure handling mechanisms, see for instance (Georgeff, Lansky, & Schoppers 1986; Simmons 1990). Nevertheless, so far this research has focused on domain specific applications and on system architectures. As a consequence, the current work is far for providing a general theoretical framework for representing and reasoning about failure. A partial exception is the work described in (Georgeff & Lansky 1986), where the semantics of PRS processes is constructed via a failure and a success set. A difference between MRG and PRS is that in MRG, failure and success are represented explicitly in the planning language. We have circumscribed the set of basic operations and constructs that deal with failure (i.e. iffail, Σ and Φ). Given these building blocks, it is possible to give a clear semantics to MRG expressions equivalent to PRS processes. Moreover, we have provided a logical theory (MT^*) to reason deductively about MRG plans, while PRS does not allow this.

Conclusions and future work

We have described a language for planning, called MRG, where failure and success are represented explicitly. This language allows control constructs that capture and react to failure. This opens up the possibility of building plans that recover from failure in different ways depending on different situations, e.g. either by replanning (as in classical planners) or by executing exception handling routines (as in embedded planners). This planning language has a declarative semantics that provides a formal explanation to failure and success and that can be used to describe actual executions within the real world. Finally, we have described a logical metatheory where it is possible to reason (hypothetically) about failure and success.

At the moment, MRG is the planning language used in a large scale, real world application under development at IRST (Stringa 1991; Spalazzi, Cimatti, & Traverso 1992b). This application aims at the development of

a system that has to control and coordinate mobile robots, navigating in unpredictable enviroments inhabited by humans and performing high level tasks, like transportation tasks in hospitals and offices (Stringa 1991; FIRST 1993). MRG plans are executed by means of systems that act in the real world, e.g. a reactive sensor and actuator controller for navigation tasks and a system for speech recognition.

Major future goals include extensions to MT^* to reason about the whole set of MRG plans, in particular the plans that contain representations of the internal activities of the planner itself, e.g. plan generation, plan execution and information acquisition (Traverso, Cimatti, & Spalazzi 1992). We also intend to develop a complete formal account of the relations between MRG and MT^*. The idea is to extend the results proved for a logical metatheory for theorem proving (Giunchiglia & Traverso 1991; 1994) which is expressive enough to represent theorem proving strategies and failure.

Acknowledgments

This work has been conducted as part of MAIA, the integrated AI project under development at IRST. A preliminary version of the work described in this paper has been presented and discussed at the Dagstuhl seminar on "Deductive Approaches to Plan Generation and Plan Recognition", 1993. We thank the members of the Mechanized Reasoning Groups in Trento and Genoa. We also thank Michael Georgeff and Sam Steel for fruitful discussions about this work.

References

Allen, J., and Ferguson, G. Actions and Events in Interval Temporal Logic. Submitted to J. Logic and Computation, Special Issue on Action and Processes.

Beetz, M., and McDermott, D. 1992. Declarative Goals in Reactive Plans. In Hendler, J., ed., *Artificial Intelligence Planning Systems: Proc. of 1st International Conference*, 3–12. San Mateo, CA: Morgan Kaufmann.

Cimatti, A.; Traverso, P.; Dalbosco, S.; and Armando, A. 1992. Navigation by Combining Reactivity and Planning. In *Proc. Intelligent Vehicles '92*. IRST-Technical Report 9205-11, IRST, Trento, Italy.

Firby, R. J. 1992. Building Symbolic Primitives with Continuous Control Routines. In Hendler, J., ed., *Artificial Intelligence Planning Systems: Proc. of 1st International Conference*, 62–69. San Mateo, CA: Morgan Kaufmann.

FIRST. 1993. Friendly Interactive Robot for Service Tasks. EUREKA EU474 PROJECT.

Georgeff, M., and Lansky, A. L. 1986. Procedural knowledge. *Proc. of IEEE* 74(10):1383–1398.

Georgeff, M.; Lansky, A.; and Schoppers, M. 1986. Reasoning and planning in dynamic domains: An experiment with a mobile robot. Technical Report 380, A.I. Center, SRI International.

Giunchiglia, F., and Traverso, P. 1991. Reflective reasoning with and between a declarative metatheory and the implementation code. In *Proc. of the 12th International Joint Conference on Artificial Intelligence*, 111–117. Also IRST-Technical Report 9012-03, IRST, Trento, Italy.

Giunchiglia, F., and Traverso, P. 1994. Program Tactics and Logic Tactics. In *Proceedings 5th Intnl. Conference on Logic Programming and Automated Reasoning (LPAR'94)*. Also IRST-Technical Report 9301-01, IRST, Trento, Italy. Presented at the Third International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida, January 1994.

Harel, D. 1984. Dynamic Logic. In Gabbay, D., and Guenthner, F., eds., *Handbook of Philosophical Logic*, volume II. D. Reidel Publishing Company. 497–604.

Lifschitz, V. 1993. A Language for Describing Actions. In *Proceedings Second Symposium on Logical Formalizations of Commonsense Reasoning.*

Rosenschein, S. 1981. Plan synthesis: A logical perspective. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, 331–337.

Simmons, R. 1990. An Architecture for Coordinating Planning, Sensing and Action. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, 292–297.

Simmons, R. 1991. Coordinating planning, perception and action for mobile robots. In *AAAI Spring Symposium on Integrated Intelligent Architectures*, 156–159. SIGART Bulletin vol. 2 no. 4.

Spalazzi, L.; Cimatti, A.; and Traverso, P. 1992. Implementing planning as tactical reasoning. In *Proc. AIS'92, AI Simulation and Planning in High Autonomy Systems Conference*, 80–85. Perth Australia: IEEE Computer Society Press.

Steel, S. 1993. Actions on Beliefs. To appear in: Journal of Applied Non-Classical Logics.

Stringa, L. 1991. An Integrated Approach to Artificial Intelligence: the MAIA Project at IRST. Technical Report 9103-13, IRST, Trento, Italy.

Traverso, P., and Spalazzi, L. 1993. Towards a theory of embedded planning. Technical Report 9308-01, IRST, Trento, Italy.

Traverso, P.; Cimatti, A.; and Spalazzi, L. 1992. Beyond the single planning paradigm: introspective planning. In *Proceedings ECAI-92*, 643–647. IRST-Technical Report 9204-05, IRST, Trento, Italy.

Wilkins, D. 1985. Recovering from execution errors in SIPE. *Computational Intelligence* 1:33–45.