Towards A Formal Theory of Repair in Plan Execution and Plan Recognition

David R. Traum and James F. Allen
Computer Science Department
University of Rochester
Rochester, New York 14627-0226 USA
{traum,james}@cs.rochester.edu

Abstract

We present a situation theoretic formalization of plan execution which allows for an abstract characterization of the role an action performance plays in the execution of a plan, including characterizations of performance error and plan repair. The *Plan Execution Situation* presented generalizes the mental state of *having a plan* to include cases where the plan is in the midst of execution, and allows for representation of dynamic change of the plan's recipe as well as the attitudes of the agent towards previous execution. We also show how this formalism can be used in different plan inference tasks such as plan execution monitoring and plan recognition.

Content Areas: intelligent agency, formal models

1 Introduction

Error and repair are essential notions for reasoning about action in a complex and uncertain environment. The topic of repair in natural language conversation has been a topic of much study in Conversation Analysis and Psycholinguistics (e.g. [Schegloff *et al.*, 1977; Levelt, 1983; Clark and Schaefer, 1989]), and more recently in AI (e.g. [Litman and Allen, 1990; Raudaskoski, 1990; McRoy, 1993; Traum and Hinkelman, 1992]). There has also been work in plan execution monitoring and replanning in an uncertain world (e.g. [Peng Si Ow, 1988; Ambros-Ingerson and Steel, 1988]), but there has as yet been little foundational work on explicating formally just what errors and repairs *are*, and how they relate to the task of performing a plan. These concepts are necessary for reasoning about any plan execution domain in which performance errors or dynamic replanning may take place.

Pollack and others have mentioned the need for distinguishing between plans as "recipes" for actions and the mental state of *having a plan*. She presents a definition of having a *simple plan* [Pollack, 1986; Pollack, 1990] in terms of beliefs and intentions that an agent has about the recipe's constituent actions. This formulation was modified by Grosz and Sidner to represent a *Shared Plan* between two agents [Grosz and Sidner, 1990]. While these formulations represent a significant increase in sophistication over previous frameworks for intended plan recognition, they are still inadequate for modelling a common plan inference situation - one in which the agent has begun executing the plan. One of the requirements for having a simple or shared plan is that the agent(s) intend all of the actions in a plan. But consider a case in which the first action has already been performed. Clearly the agent does not intend this action any more, yet it is still a part of the plan.

One might try to amend such a theory by claiming that the intended plan *plan* consists of just those parts of the original recipe which have not yet been performed (and about which agents still have intentions). This formulation would have a number of deficiencies, however. First, it would not allow for a convenient analysis of repair, since agents often repair actions which have already been performed. Also, it will not allow represenation of the roles an action plays towards plan execution, since it would be impossible to distinguish the continuation of an ongoing plan from the initiation of a plan.

It is often useful to be able to characterize, in an abstract sense, what role a particular performance plays in the execution of a plan. In cases of intended recognition (in which the performing agent intends for the observing agent to recognize her plan) such as natural language communication, the performing agent often gives signals as to how the current action relates to plans which the observing agent should infer: whether the current performance begins a new plan, or continues, completes, repairs or cancels a pre-existing plan. For example, uttering the English clue word "oops" indicates a belief that a previous action failed. Similarly, researchers have found prosodic markings for repairs [Levelt and Cutler, 1983] and nonfinality [McLemore, 1991].

1.1 An Informal Example

We will illustrate this formalism with the following example, taken from Kautz' cooking microworld [Kautz, 1987]. Consider recognizing the actions of a somewhat inept cook. The cook decides to make a meal, and after some deliberaton decides to make Spaghetti Marinara. The first action the cook performs is to make the marinara sauce, which, to an observer could be a step in making either Spaghetti Marinara or Chicken Marinara. The next thing the cook does is make Spaghetti, disambiguating the intent to make Spaghetti Marinara. Next, the cook starts to boil the noodles. Our cook, however boils the noodles too long, resulting in a pasty soup. Now the cook has some sort of repair to make, and there are several choices of next action. The cook could (1) just ignore the previous boiling, and boil some more noodles, assuming there is another pot and more noodles. If not, (2) the cook might have to perform some other actions such as cleaning the pot and/or making more noodles. A third alternative is to give up on the Spaghetti and make some Chicken, deciding on a meal of Chicken Marinara instead.

While in each of the three scenarios, all of the actions described above are performed in service to the agent's desire and evolving plan to prepare a meal, there is no single meal recipe that contains all of these actions, and thus a system such as Kautz's could not recognize this activity as fitting into a single event. The best that it could do would be to decide that the cook was preparing two meals (perhaps also with other events such as washing dishes), one of which was uncompleted.

2 Sketch of Rational, Plan-based Behavior

Before proceeding with a formal account of plan execution, it will be helpful to informally present a sketch of the mental attitudes that a plan-executing agent will have and the deliberative processes that it will undergo. We choose a model similar to the BDI model of [Bratman *et al.*, 1988].

We start with **beliefs** and **desires**. From the desires, (and based on beliefs and other goals and intentions), the agent will *deliberate* and choose a set of **goals**: conditions that the agent will try to achieve. *Planning* or *means-ends reasoning* will lead to the formation or selection of a **plan recipe** designed to meet the goals. We view recipes as consisting of a set of actions coupled with a set of constraints relating various properties of these actions (e.g. constraints on relative timing or objects and locations of actions, preconditions or effects represented as events or states which must hold over times related to the times of actions). The agent then can *adopt* or *commit to* plans. **Plans** are treated as individuals which can be modified through time. At any given moment in time, a particular plan will correspond to a particular plan recipe. This plan adoption will lead to **commitment** to achieve (or maintain) the constraints of the plan and **intention** to perform the actions in the plan.

When an agent has committed to a plan, she can *try* to perform an action in that plan. She can also *observe* the situation she finds herself in, perhaps revising her beliefs and desires. She can also *replan*, revising the plans she is executing to correspond to different plan recipes, and thus changing the her commitments and intentions. We distinguish two types of plan revision, *plan elaboration*, in which additional actions (e.g., decompositions of non-primitive actions) or constraints are added to a plan and *plan repair*, in which some of the actions or constraints are removed. We can also distinguish the action of *plan repair* from *changing plans*. In both cases, the intentions and commitments of an agent change, but in the former case, the agent is changing the contents of a plan which she continues to execute, whereas in the latter, the agent drops all intentions and adopts a new set.

From the point at which a plan is adopted to the point at which the intentions and commitments are dropped, we say that the agent is *executing* the plan.

We can distinguish at least three distinct notions of the culmination of plan:

Action Completion – all actions in the plan have been performed.

Successful Completion – all actions in the plan have been performed and all of the constraints have been met as well.

Goal Satisfaction – the goals which motivated adoption of the plan have been achieved.

Note that Successful Completion and Goal Satisfaction are somewhat independent. It may be the case that the goals are met before the plan is completed (e.g., imagine a plan to open an elevator door by pushing a button: the goal is to get the elevator door open, but suppose it opens by itself when someone else gets out). Depending on the plan adoption procedures, a plan might also be successfully completed without having satisfied the goals. A plan is an independent entity from the goals for which it was adopted, and the same plan could be used to try to achieve many different sets of goals. If a goal is not a constraint of the plan, it may be the case that the plan is successfully executed but the goals are not met. In a distributed control scenario, a plan executor may not have access to actual goals and may simply adopt plans under orders of a superior.

A plan executor will monitor the success of the plans it is executing, and engage in a repair if the plan is not successful. But the plan executor will also need to monitor goals: plan repair might also be warranted if the situation changes. This could include eliminating unnecessary actions if the goals get met though events external to the plan itself.

3 Basic Ontology

We start with a Situation Theory, similar to [Devlin, 1991]. Situations are individuals, and also provide bases for the truth or falsity of propositions. We will notate general situations by the letter S, perhaps subscripted. For situation S and proposition ϕ the notation "S $\models \phi$" means that the proposition ϕ is supported by situation S. Situations are also composable into other situations, forming a lattice structure. Following Devlin we overload the operator "$\subseteq$" to include the "part of" relation between two situations as well as the usual subset relation between two sets. $S_1 \subseteq S_2$ means that situation S_1 is a part of situation S_2.

An interesting class of situations will be those corresponding to "happenings", which we will term *occurrences* and notate with lowercase Greek letters. A subclass of these, those occurrences which are *caused* by the intentional activity of agents will be termed *executions*. The key point of an execution is not that it has particular properties intended by a performing agent, but that intentional activity was instrumental in the performance.

Also important are abstractions over executions. We will term these *actions*, and it is these actions which are components of mentalistic attitudes such as intentions and plans. While a particular execution has many features, only a small subset of them are actually intended. What we call *actions* are sometimes called *action types* [Goldman, 1970] or *activities* [Balkanski, 1990].

We use the symbol "$\triangleright$" to represent the "realizes" or "is characterized by" relation between an execution and an action. An execution could realize unrelated action types, and whenever an execution realizes an action type, it always realizes a more abstract action type as well.

As an example, suppose there is some execution α which realizes a MakeSpaghetti action: $\alpha \rhd \mathrm{MakeSpaghetti(Agt, time)}$. It would then also be the case that this same execution realizes a MakeNoodles action:
$\alpha \rhd \mathrm{MakeNoodles(Agt, time)}$. It might also be the case that this same execution also realizes some unrelated action, say $\alpha \rhd \mathrm{WakeUpDog(Agt, time)}$. In this case, we have two separate actions performed in the same execution.

3.1 Plans and Mental Attitudes

We treat plans as individuals; they are abstract objects whose attributes can be modified through time, just as a physical objects can. We use the term *recipe* to denote the functional characteristics of a plan at a particular time. A recipe is a set of actions coupled with a set of constraints relating various properties of these actions (e.g., constraints on relative timing, objects and locations of actions, goals and preconditions represented as events or states which must hold over times related to the times of actions). We denote the set of actions of a recipe, R, by $\mathbf{Actions}(\mathrm{R})$. The set of constraints on a recipe will be denoted by $\mathbf{Constraints}(\mathrm{R})$[1].

The agent of an action, $\mathrm{a_i}$, in a recipe, R, is denoted by $\mathbf{Agt}(\mathrm{a_i}, \mathrm{R})$. For single agent recipes, all actions in the recipe will have the same agent parameter.

A recipe can be said to have been performed in a situation if all the actions have been performed and all of the constraints have been met. Formally,

Definition 1 $\mathbf{Performedin}(\mathrm{R}, \mathrm{S})$ *iff*

$$\forall \mathrm{a_i} : \mathrm{a_i} \in \mathbf{Actions}(\mathrm{R}) \supset \exists \alpha : \alpha \subseteq S \wedge \alpha \rhd \mathrm{a_i} \;\; \wedge$$
$$\forall C : C \in \mathbf{Constraints}(\mathrm{R}) \supset S \models C$$

A plan may be reflected by different recipes at different times, changing as an agent modifies the plan. We denote the recipe of a plan, P, at time t as $\mathbf{Recipe}(\mathrm{P}, \mathrm{t})$. As a shorthand, the actions of a plan, $\mathbf{Actions}(\mathbf{Recipe}(\mathrm{P}, \mathrm{t}))$. will be denoted $\mathbf{Actions}(\mathrm{P}, \mathrm{t})$. The set of constraints on a plan will be denoted by $\mathbf{Constraints}(\mathrm{P}, \mathrm{t})$.

There are several other mentalistic notions which we will assume, but not attempt to define or axiomatize. These are: belief, intention, and intentional-action. Belief is a relation between an agent and a proposition,

[1] Many of these constraints will be implicit in any *Representation* of a recipe, e.g. as shared variables in two separate actions, or domain constraints on possible recipes

and we will use the notation[2], $\mathbf{Bel}(\mathrm{Agt}, \phi)$, to represent that agent **Agt** believes ϕ. We represent (future directed) intention as having two arguments other than the agent, an action which the agent intends to perform, and a plan which this action is intended to (in part) achieve, $\mathbf{Intends}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$. This is roughly equivalent the expression Int(Agt,By(a,P)) used by Pollack and Grosz & Sidner. Finally, we represent present directed intention as an abstract action in its own right, called *try*. $\alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$ means that the occurrence α is characterized by **Agt** intending to do action **a** as part of executing Plan **P**.

We can now formally define an execution as an occurrence for which there is some agent and plan such that the occurance realizes the agent trying to perform an action in the plan: α is an execution iff $\exists Agt, a, P$: $\alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P})$

3.2 Plan Execution Situations

Now we are in a position to introduce the central theoretical construct of this paper, the Plan Execution Situation, roughly a generalization of Pollack's notion of an agent *having* a plan [Pollack, 1986; Pollack, 1990], translated to the logic of situations.

A *Plan Execution Situation* (PES) is a situation in which a plan is being executed. It is a piece of the world containing the relevant agents, objects, and events that make up executions or attempted executions of plans. We represent a plan execution situation PE as a 5-tuple,
$[\mathrm{Agt}, \mathrm{P_{PE}}, \mathrm{E_{PE}}, \mathrm{Bind_{PE}}, \mathrm{S_{PE}}]$ where:

Agt represents the agent that is executing the plan.

$\mathrm{P_{PE}}$ represents the plan that this is the execution *of*.

$\mathrm{E_{PE}}$ represents the set of executions which have been performed. in executing the plan. I.e., $\alpha \in \mathrm{E_{PE}}$ iff $\alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P_{PE}})$.

$\mathrm{Bind_{PE}}$ represents the *instantiation* (partial) function from actions of the plan to performed executions. For each action in the plan, its instantiation is an execution which the agent believes realizes that action. We will generally view this function as a set of relationships of the form: $\mathrm{a_i} \rightsquigarrow \alpha_\mathrm{i}$ where $\mathrm{a_i} \in \mathbf{Actions}(\mathrm{P_{PE}})$, $\alpha_\mathrm{i} \in \mathrm{E_{PE}}$ We will use a superscript version, $\mathrm{a_i} \overset{\mathrm{PE}}{\rightsquigarrow} \alpha_\mathrm{i}$ as shorthand to indicate that $\mathrm{a_i} \rightsquigarrow \alpha_\mathrm{i} \in \mathrm{Bind_{PE}}$.

[2] For simplicity, we omit the situational and temporal arguments of belief and intention.

S_{PE} represents the execution status, either 1 or 0, representing whether or not the plan is being executed. In this paper we will only consider PES's with status 1.

Each PES will also have a designated current time, **now**, and so we will generally omit the temporal arguments from the recipe designators for the PES' plan. Thus $\mathbf{Actions}(P_{PE})$, $\mathbf{Constraints}(P_{PE})$, and $\mathbf{Recipe}(P_{PE})$ will be used to represent the components of a plan at the **now** time of the PES. Similarly, $\mathbf{Agt}(a_i, P_{PE})$ and $\mathbf{Time}(a_i, P_{PE})$ will be used to represent the agent and time of an action in the Plan's recipe at the **now** time of the plan execution situation.

Two predicates will be useful for classifying actions in a PES's plan as to whether they have been performed or not:

Definition 2 $\mathbf{Instantiated}(a, PE)$ *iff* $a \in \mathbf{Actions}(P_{PE}) \wedge \exists_{\alpha \in E_{PE}} a \overset{PE}{\rightsquigarrow} \alpha$

Definition 3 $\mathbf{Uninstantiated}(a, PE)$ *iff* $a \in \mathbf{Actions}(P_{PE}) \wedge$
$\neg\mathbf{Instantiated}(a, PE)$

Note that an action that is not in the plan at all is neither instantiated or uninstantiated with respect to the PES.

An agent is executing a plan if (1) the agent intends to perform all uninstantiated actions in support of the plan and (2) for each instantiated action, the agent believes that the execution which the action is instantiated to realizes that action. With regards to particular situations, an agent, Agt, is executing a plan P_{PE} in a situation S iff there is a PES with status 1 that is part of situation S. A PES is part of a situation if all the executions of the PES are part of the situation and resulted from attempts to perform actions in the plan and the agent intends to perform all the uninstantiated actions, and believes that the instantiated actions are realized by the executions to which they are bound. Formally,

Axiom 1 $\forall PE, S : PE \subseteq S \equiv$

$$\begin{array}{l} \forall_{\alpha \in E_{PE}} \alpha \subseteq S \ \wedge \\ \forall [a_i \rightsquigarrow \alpha_i \in Bind_{PE}] S \models \mathbf{Bel}(Agt, \alpha_i \rhd a_i) \ \wedge \\ \qquad (S_{PE} = 1) \wedge \forall a : \mathbf{Uninstantiated}(a, PE) \supset S \models \mathbf{Intends}(Agt, a, P_{PE}) \vee \\ \qquad (S_{PE} = 0) \wedge \neg \exists a : S \models \mathbf{Intends}(Agt, a, P_{PE}) \end{array}$$

We define the remaining acts of a plan execution situation, $\mathbf{RActs}(PE)$ as those actions in the plan which have not been instantiated by any of the executions – these are the actions which still remain to be executed in order for the plan execution situation to be completed.

Formally, $\mathbf{RActs}(\mathrm{PE}) \stackrel{def}{=} \{\mathrm{a_i} \mid \mathbf{Uninstantiated}(\mathrm{a_i}, \mathrm{PE})\}$

A Plan Execution Situation, PE is **action completed** if all of the actions in the plan are instantiated by executions. Formally,
$\mathbf{ACompleted}(\mathrm{PE})$ iff $\mathbf{RActs}(\mathrm{PE}) = \{\}$

A PES is **successful** in a situation if it is completed and all of its constraints have been met: $\mathbf{Successful}(\mathrm{PE}, \mathrm{S})$ iff
$\mathbf{ACompleted}(\mathrm{PE}) \wedge PE \subseteq S \wedge \forall C : C \in \mathbf{Constraints}(\mathrm{P_{PE}}) \supset S \models C$

We call a plan execution situation *feasible* in a given situation if performances of the uninstantiated actions will lead to a situation in which the plan execution is successful (minimizing unexpected external events).

4 Updating Plan Execution Situations

The *updating* of a situation by an occurrence is a situation that includes both the old situation as well as the occurrence. Generally this will involve adding an occurrence which happens at the **now** point of the situation, and the **now** point is then updated to immediately after the time of the occurrence. We introduce a situation updating function, $\mathbf{Update}(\alpha, \mathrm{S})$ which takes as arguments an occurrence and a situation, and returns the updated situation. A specialization of this is when the occurrence is an execution relevant to a plan execution situation.

There are several ways in which an execution α may update a plan execution situation PE. The simplest is if the execution is intended to instantiate one or more of the actions in $\mathbf{RActs}(\mathrm{PE})$. We define a predicate **DoNext** over an execution and a PES which is true when there is a second PES which is an update of the first with the execution and binding relationship added:

Definition 4 $\mathbf{DoNext}(\alpha, \mathrm{PE})$ *iff*[3]

$$\begin{aligned} &\exists \mathrm{a} \in \mathbf{RActs}(\mathrm{PE}) : \alpha \triangleright \mathbf{Try}(\mathrm{Agt}, \mathrm{a}, \mathrm{P_{PE}}) \wedge \\ &\quad \exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge \mathbf{Recipe}(\mathrm{P_{PE'}}) = \mathbf{Recipe}(\mathrm{P_{PE}}) \wedge \\ &\qquad \mathrm{E_{PE'}} = \mathrm{E_{PE}} \cup \{\alpha\} \wedge \mathrm{Bind_{PE'}} = \mathrm{Bind_{PE}} \cup \{\mathrm{a} \rightsquigarrow \alpha\} \end{aligned}$$

[3]For simplicity, the definition presented here allows an execution to realize only a single action in the plan. The actual definition replaces the bound action, a, with a subset of $\mathbf{RActs}(\mathrm{PE})$

4.1 Sub-Action Relations

We may further classify executions as to what role they play in the progression of the plan execution. The first execution of an action in the plan of a PES will *begin* the execution of the plan:

Definition 5 $\mathbf{BEGINS}(\alpha, \mathrm{PE})$ *iff* $\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathrm{E_{PE}} = \{\}$

Subsequent executions will *continue* the plan execution:

Definition 6 $\mathbf{CONTINUES}(\alpha, \mathrm{PE})$ *iff* $\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathrm{E_{PE}} \neq \{\}$

The final execution will *complete* the plan execution:

Definition 7 $\mathbf{COMPLETES}(\alpha, \mathrm{PE})$ *iff*
$\mathbf{DoNext}(\alpha, \mathrm{PE}) \wedge \mathbf{ACompleted}(\mathbf{Update}(\alpha, \mathrm{PE}))$

An execution might also instantiate all actions in the plan:

Definition 8 $\mathbf{PERFORMS}(\alpha, \mathrm{PE})$ *iff*
$\mathbf{BEGINS}(\alpha, \mathrm{PE}) \wedge \mathbf{COMPLETES}(\alpha, \mathrm{PE})$

4.2 Failure and Repair

We can define a *failure* as an execution which does not realize the action which it was intended to.

Definition 9 $\mathbf{Fail}(\alpha)$ *iff* $\exists \mathrm{Agt, a, P} : \alpha \rhd \mathbf{Try}(\mathrm{Agt, a, P}) \wedge \neg\alpha \rhd \mathrm{a}$

What is more relevant for plan execution than a normative characterization of failure is perceived failure. That is, an execution which the performing agent takes to have been a failure. These are any executions of a PES which are not bound to an action in the plan.

Definition 10 $\mathbf{PerFail}(\alpha, \mathrm{PE})$ *iff*
$\alpha \in \mathrm{E_{PE}} \wedge \neg\exists \mathrm{a_i} : (\mathrm{a_i} \in \mathbf{Actions}(\mathrm{P_{PE}}) \wedge \mathrm{a_i} \overset{\mathrm{PE}}{\leadsto} \alpha)$

If an agent believes that her action is a failure, the updated PES will include the execution, but will not have any new binding relationships.

There are at least two ways of continuing a plan execution other than simply updating the plan by performing a next action. One is to change the instantiation function, such that either some action $\mathrm{a_i}$ in the plan which was previously instantiated by some some execution α is no longer instantiated by that execution, thus requiring a new execution to bind that action in any completed plan, or, conversely, that some a_i which was previously uninstantiated is newly bound by an execution α already part of the PES. Other instantiations in the Plan execution would remain the same. We call this **ERepair**, standing for Execution Repair, since the same plan is maintained but the beliefs are changed about the success of previous action, and thus intentions about future actions are also changed.

Definition 11 $\mathbf{ERepair}(\alpha, \mathrm{PE})$ *iff* $\exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge$
$\mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}'}) = \mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}}) \wedge \mathrm{E}_{\mathrm{PE}'} = \mathrm{E}_{\mathrm{PE}} \wedge \mathrm{Bind}_{\mathrm{PE}'} \neq \mathrm{Bind}_{\mathrm{PE}}$

Another type of plan execution repair is to modify the plan itself, so that it is composed of a different recipe, though it still uses some of the same executions. This might also entail further **ERepair**, if it eliminates actions which have already been instantiated. We call this **PRepair**, standing for Plan Repair, since we are changing the plan that is being executed. In a **PRepair** the agent has changed intentions from performing the actions in the old recipe to performing actions in the new one.

Definition 12 $\mathbf{PRepair}(\alpha, \mathrm{PE})$ *iff*
$\exists \mathrm{PE}' : \mathrm{PE}' = \mathbf{Update}(\alpha, \mathrm{PE}) \wedge \mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}'}) \neq \mathbf{Recipe}(\mathrm{P}_{\mathrm{PE}}) \wedge \mathrm{E}_{\mathrm{PE}'} = \mathrm{E}_{\mathrm{PE}}$

Some occurrences will be relevant for the eventual success or failure of a plan execution, but do not fall into any of the above categories. These will not be part of the PES as such, but will certainly be included in updates of the situation in which success is evaluated, and may be cause for an agent to undertake plan repair, especially if they affect the feasibility of the plan execution.

4.3 Example

We now return to the cooking example presented at the beginning of the paper. The plan recipe for Spaghetti Marinara might have the following set of actions[4]: {a_1:MakeSpaghetti, a_2:MakeMarinara, a_3:BoilNoodles, a_4:AssembleMeal}, as well as an appropriate set of constraints on execution order and the form of the products resulting from each action. Once the

[4] Ignoring for now specializations and abstractions of these actions.

agent has adopted this plan, we will have PES PE_0 as part of the current situation, where $\mathbf{Actions}(\mathrm{P}_{\mathrm{PE}_0}) = \{\mathrm{a}_1,\mathrm{a}_2,\mathrm{a}_3,\mathrm{a}_4\}$, and the execution and binding relationship sets are empty. $\mathbf{RActs}(\mathrm{PE}_0) = \{\mathrm{a}_1,\mathrm{a}_2,\mathrm{a}_3,\mathrm{a}_4\}$, and PE_0 is feasible, since there is a sequence of possible executions which will successfully execute the plan. The agent **begins** execution by performing α_1, making the sauce (i.e. $\alpha_1 \ \triangleright$MakeSauce). The result is: $\mathrm{PE}_1 = \mathbf{Update}(\alpha_1, \mathrm{PE}_0) = [\mathrm{Agt}, \mathrm{P}, \{\alpha_1\}, \{\mathrm{a}_2 \rightsquigarrow \alpha_1\}, 1]$. The next execution, α_2 is making the spaghetti, and so: $\mathrm{PE}_2 = \mathbf{Update}(\alpha_2, \mathrm{PE}_1) = [\mathrm{Agt}, \mathrm{P}, \{\alpha_1, \alpha_2\}, \{\mathrm{a}_2 \rightsquigarrow \alpha_1, \mathrm{a}_1 \rightsquigarrow \alpha_2\}, 1]$. Next we get α_3 the overboiling of the noodles. This leads initially to $\mathrm{PE}_3 = \mathbf{Update}(\alpha_3, \mathrm{PE}_2) = [\mathrm{Agt}, \mathrm{P}, \{\alpha_1, \alpha_2, \alpha_3\}, \{\mathrm{a}_2 \rightsquigarrow \alpha_1, \mathrm{a}_1 \rightsquigarrow \alpha_2, \mathrm{a}_3 \rightsquigarrow \alpha_3\}, 1]$. However this plan execution is infeasible, since the overboiled noodles will not be usable. The agent notices this, deciding α_3 is a failure, and performs an **ERepair**. This will be α_4, and the resulting PES is $\mathrm{PE}_4 = \mathbf{Update}(\alpha_4, \mathrm{PE}_3) = [\mathrm{Agt}, \mathrm{P}, \{\alpha_1, \alpha_2, \alpha_3\}, \{\mathrm{a}_2 \rightsquigarrow \alpha_1, \mathrm{a}_1 \rightsquigarrow \alpha_2\}, 1]$. Note that α_4, itself, is not in the execution list, since it was not done with the intention to perform an action in the plan. Also, the binding relationship from a_3 to α_3 has been removed, but α_3 remains in the set of executions. It thus follows that $\mathbf{PerFail}(\alpha_3, \mathrm{PE}_4)$. Now taking the simplest case, the agent just continues by boiling more noodles (α_5), and then completes the plan by assembling the meal (α_6). The final PES will be $\mathrm{PE}_6 = \mathbf{Update}(\alpha_6, \mathbf{Update}(\alpha_5, \mathrm{PE}_4))$ $= [\mathrm{Agt}, \mathrm{P}, \{\alpha_1, \alpha_2, \alpha_3, \alpha_5, \alpha_6\}, \{\mathrm{a}_2 \rightsquigarrow \alpha_1, \mathrm{a}_1 \rightsquigarrow \alpha_2, \mathrm{a}_3 \rightsquigarrow \alpha_5, \mathrm{a}_4 \rightsquigarrow \alpha_6\}, 1]$.

Plan recongition proceeds in much the same way as the execution sequence described above. Initially the recognizer will not know what plan is being executed. On observing α_1, the recognizer will assume a plan to make either Spaghetti Marinara or Chicken Marinara, as with Kautz's system. Observation of α_2 will disambiguate the plan, and the recognizer will now expect the other two actions. On noticing the overboiling, α_3, the recognizer will realize the plan will not succeed, and will be primed for a repair of some sort. Observation of the second boiling, α_5, can be taken to be a re-execution, leading to a hypothesis that an **ERepair** was performed, α_4. Thus, unlike Kautz's system, this formalism allows the representation of a hypothesis that all the observed occurrences were performed in support of a single plan, rather than multiple events. Space does not permit going over the other examples, but they proceed in much the same way.

5 Conclusion

The formulation given here provides the basis for a formal theory of plan execution that can be used uniformly for a variety of plan inference tasks, including plan recognition, plan execution monitoring, and interleaved planning and execution. It naturally extends mentalistic representations of *having* a plan to include cases where execution is underway. A natural semantics for repair is provided which allows extension of plan recognition techniques to be able to represent cases of failed execution and dynamic plan change.

While the formalism given above does little to advance the practice of plan execution monitoring, it provides a formal theory which can be easily integrated with other plan inference tasks. In cases of intended plan recognition, such as natural language communication, the executing agent often provides extra information about how a performance relates to other actions in the plan. The PES formalism provides a convenient way of reasoning about the execution state so that the proper signals may be given and interpreted.

References

[Ambros-Ingerson and Steel, 1988] J.A. Ambros-Ingerson and S. Steel, "Integrating planning, execution, and monitoring," In *Proceedings AAAI-88*, pages 83–88, 1988, Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 735–740.

[Balkanski, 1990] Cecile T. Balkanski, "Modelling Act-Type Relations In Collaborative Activity," Technical Report 23-90, Harvard University Center for Research in Computing Technology, 1990.

[Bratman *et al.*, 1988] Michael E. Bratman, David J. Israel, and Martha E. Pollack, "Plans and Resource-Bounded Practical Reasoning," Technical Report TR425R, SRI International, September 1988, Appears in Computational Intelligence, Vol. 4, No. 4, 1988.

[Clark and Schaefer, 1989] Herbert H. Clark and Edward F. Schaefer, "Contributing to Discourse," *Cognitive Science*, 13:259 – 94, 1989.

[Devlin, 1991] Keith Devlin, *Logic and Information*, Cambridge University Press, 1991.

[Goldman, 1970] Alvin I. Goldman, *A Theory of Human Action*, Prentice Hall Inc., 1970.

[Grosz and Sidner, 1990] Barbara J. Grosz and Candace L. Sidner, "Plans for Discourse," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Kautz, 1987] H. A. Kautz, *A Formal Theory of Plan Recognition*, PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1987, Also available as TR 215, Department of Computer Science, University of Rochester.

[Levelt, 1983] Willem J. M. Levelt, "Monitoring and self-repair in speech," *Cognition*, 14:41–104, 1983.

[Levelt and Cutler, 1983] Willem J. M. Levelt and A. Cutler, "Prosodic Markings in Speech Repair," *Journal of Semantics*, 2:205–217, 1983.

[Litman and Allen, 1990] Diane J. Litman and James F. Allen, "Discourse Processing and Common Sense Plans," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[McLemore, 1991] Cynthia McLemore, *The Pragmatic Interpretation of English Intonation*, PhD thesis, University of Texas at Austin, 1991.

[McRoy, 1993] Susan McRoy, *Abductive Interpretation and Reinterpretation of Natural Language Utterances*, PhD thesis, University of Toronto, 1993, Reproduced as TR CSRI-288 Department of Computer Science, University of Toronto.

[Peng Si Ow, 1988] Alfred Thierez Peng Si Ow, Stephen F. Smith, "Reactive Plan Revision," In *Proceedings AAAI-88*, August 1988.

[Pollack, 1990] Martha E. Pollack, "Plans as Complex Mental Attitudes," In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Pollack, 1986] Martha E. Pollack, *Inferring Domain Plans in Question-Answering*, PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 1986.

[Raudaskoski, 1990] Pirkko Raudaskoski, "Repair Work in Human-Computer Interaction," In Paul Luff, Nigel Gilbert, and David Frohlich, editors, *Computers and Conversation.* Academic Press, 1990.

[Schegloff *et al.*, 1977] E. A. Schegloff, G. Jefferson, and H. Sacks, "The Preference for Self Correction in the Organization of Repair in Conversation," *Language*, 53:361–382, 1977.

[Traum and Hinkelman, 1992] David R. Traum and Elizabeth A. Hinkelman, "Conversation Acts in Task-oriented Spoken Dialogue," *Computational Intelligence*, 8(3):575–599, 1992, Special Issue on Non-literal language.