Parsing Plans Situation-Dependently in Dialogues

Kiyoshi Kogure, Akira Shimazu and Mikio Nakano
NTT Basic Research Laboratories
3-1 Morinosato-Wakamiya, Atsugi, Kanagawa, 243-01 Japan
{kogure, shimazu, nakano}@atom.brl.ntt.jp

Abstract

This paper describes a plan parsing method that can handle the effects and preconditions of actions and that parses plans in a manner dependent on dialogue state changes, especially on the mental state changes of dialogue participants caused by utterances. This method is based on active chart parsing and uses augmented edge structures to keep state information locally and time map management to deal with state changes. It has been implemented in Prolog and is used for plan recognition in dialogues.

1 Introduction

Dialogue understanding requires plan recognition. Many plan inference models have thus been proposed. As an approach to the computation of plan recognition from observed actions, plan parsing has been proposed by Sidner (1985) and formalized by Vilain (1990). A typical plan recipe for an action includes a sequence of subactions as its decomposition, so interpreting an action sequence in terms of plans can be seen as parsing in which observed actions correspond to lexical tokens and plan recipes correspond to grammatical rules.

Previous plan parsing methods, however, are insufficient for dialogue understanding since they do not handle the effects and preconditions of actions. These effects and preconditions are of crucial importance in reasoning about what the agent intends to do and what she presupposes. More concretely, without treating them, it is impossible (a) to describe actions in terms of their effects, (b) to capture the relationship between an action and another action that satisfies the former's preconditions to enable it, and (c) to interpret actions in a manner dependent on the dialogue state.

To solve these problems, we have developed a plan parsing method that can handle the effects and preconditions of actions and that parses plans in a manner dependent on dialogue state changes, especially on the mental state changes of dialogue participants caused by dialogue utterances. This method, in particular, makes (a)–(c) possible. The method is based on active chart parsing and uses augmented edge structures to keep state information locally and time map management (Dean and McDermott, 1987) to deal with state changes. The method is implemented in Sicstus Prolog and is applied to a dialogue understanding system (Shimazu et al., 1994).

2 Requirements for Treating Effects and Preconditions

Let us examine typical situations where the effects and preconditions of actions must be treated.

2.1 Effect-Based Action Descriptions

In describing plan recipes, it is convenient to specify an action in terms of its effects as follows:

Recipe 1

Action:	informref(S, H, Term, Prop)
Decomposition:	achieve(bel(H, P))
Effects:	belref(H, Term, Prop)
Constraints:	parameter(Term, Prop)

A description of the form 'achieve(P)' specifies the action for achieving the state where the proposition P holds. This recipe thus says that an informref action can be performed by an action that has 'bel(H, P)' as its effect. There may be many such actions. Furthermore, the action specified by 'achieve(P)' depends on the situation where P is about to be achieved. In the extreme case, if P already holds, the agent need not do anything. For example, a speaker may not perform any action to make a hearer believe a proposition if the speaker believes the hearer already believes it. If we are not permitted to use this form, we must enumerate all the actions that achieve P together with the conditions under which they do. Treating this form requires calculating the effects of actions.

2.2 Action-Enabling

Given a goal, a planning procedure searches for an action to achieve the goal (a main action). If the procedure identifies such an action with preconditions, it calls itself recursively to search for actions

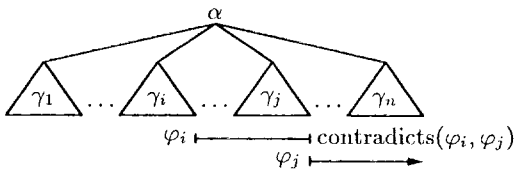

Figure 1: Effects of Complex Action.

that satisfy them (enabling actions of the main action), and then provides the action sequence consisting of the main action preceded by its enabling actions. Given an action sequence of this form, a plan recognition procedure must thus regard it as performing a main action to achieve its effect(s). There are many kinds of dialogue phenomena that can be captured by such action-enabling relationships. Understanding such dialogue phenomena requires handling effects and preconditions.

2.3 State-Dependent Interpretation

There are cases where state-dependent interpretation is impossible unless the effects and preconditions of actions are treated. Consider, for example, the following dialogue fragment:

A: Please tell me how to go to the Laboratories.
B: Take the bus to Tokyo.

Whereas an imperative sentence (with surface speech act type surface_request) is generally interpreted as a request, the second utterance actually describes a step in the plan to go to the Laboratories because the first utterance convinces B that A wants to have that plan. This latter interpretation can be captured by using the heuristic rule for selecting an interpretation with fewer unsatisfied preconditions and the following recipe:

Recipe 2

Action:	describe_step(S, H, Action, Plan)
Preconditions:	bel(S, want(H, Plan))
Decomposition:	surface_request(S, H, Action)
Constraints:	step(Action, Plan)

This interpretation would be possible instead by using a recipe whose decomposition also contains the action of making B believe A's want. However, such a recipe can handle only cases where the belief has been established by the action just before surface_request.

3 Effects and Preconditions

3.1 Effects of Actions

The effects of a linguistic action in a dialogue mainly produce unobservable mental state changes of the dialogue participants. For a computer to participate in a dialogue like people do, it must simulate such mental state changes.

The effects of an action are the propositions that hold after the action's successful execution. The effects are taken to be calculated recursively

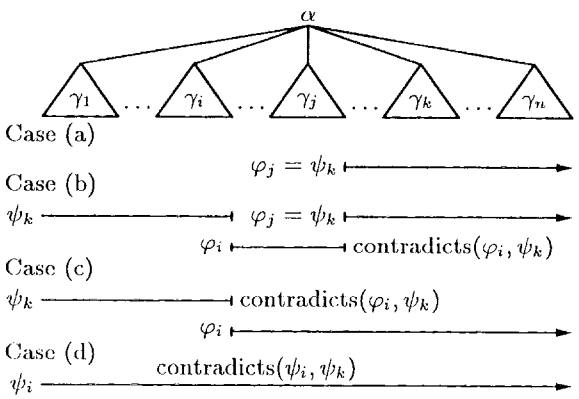

Figure 2: Preconditions of Complex Action.

from the action's recipe and component actions if any: the effects are essentially those specified by the action's recipe, plus those of component actions. Since an action is modeled to have a certain temporal extent, an action's effect is modeled to hold at the point in time where the action has just finished and to continue to persist infinitely or until the first instance that a contradictory fact holds. An effect of an action's component action also holds in the same way. Therefore, an action α with $\langle\gamma_1, \ldots, \gamma_n\rangle$ as its component actions has component action γ_i's effect φ_i as its own effect if there is no component action γ_j after γ_i with an effect φ_j contradictory to φ_i—written as contradicts(φ_i, φ_j)—and does not if such γ_j exists as in Figure 1.

3.2 Preconditions of Actions

The preconditions of an action are the propositions that must hold before the action's successful execution. Recognizing an action thus requires that its preconditions can be assured or at least hypothesized to be believed by the agent.

The preconditions of an action are essentially taken to consists of those specified by the action's recipe and those of its component actions if any. A component action's precondition, however, can be satisfied by another component action's effect. Consider action α with its component actions $\langle\gamma_1, \ldots, \gamma_n\rangle$, as shown in Figure 2. Let us focus on precondition ψ_k of action γ_k. When there is an action γ_j before γ_k such that its effect φ_j is identical to ψ_k as in Case (a) in the figure, ψ_k is satisfied by φ_j, so ψ_k need not hold at α's starting time. That is, α does not have ψ_k as its precondition. On the contrary, when there is an action γ_i before γ_k such that its effect φ_i contradicts ψ_k, ψ_k's holding at α's starting time cannot contribute to the satisfaction of γ_k's precondition ψ_k. If there exists an action γ_j between γ_i and γ_k with its effect φ_j identical to ψ_k, ψ_k can be satisfied [Case (b)]. Otherwise, ψ_k cannot be satisfied [Case (c)], so α cannot be successfully executed and should not be recognized. This kind of

interference is hereafter called 'effect-precondition (E-P) conflict.' There is another kind of interference called 'precondition-precondition (P-P) conflict:' if a precondition specified by α's recipe, or a precondition ψ_i of any other component action γ_i contradicts ψ_k, they cannot hold simultaneously at α's starting time [Case (d)]. In such a case, α should not be recognized.

4 Active Chart Plan Parsing

4.1 Decomposition Grammar

The relationship between an action and its decomposition specified by a recipe can be viewed as a phrase structure rule. The decomposition relationship specified by Recipe 2, for example, can be view as

describe_step(S, H, Action, Plan)
$\longrightarrow$ surface_request(S, H, Action).

This interpretation of the decomposition relationships specified by recipes in a plan library gives us a decomposition grammar and allows us to apply syntactic parsing techniques to plan recognition.

Based on this idea, we constructed a plan parsing method that handles the effects and preconditions of actions. Hereafter, we focus on bottom-up active chart parsing, although the core of the discussion below can be applied to other parsing methods.

4.2 Calculating Effects and Preconditions

Time Map Management

Time map management is used to capture the temporal state changes caused by the effects of actions. A time map consists of a set of (potential) fact tokens.[1] A fact token is a triple $\langle t_1, t_2, \varphi \rangle$, where t_1 and t_2 are time points and φ is a timeless fact description (a term), that represents the proposition that φ holds at t_1 and continues to persist through t_2 or until a contradictory fact holds. As a time point, we use a vertex in a chart, which is an integer. As a special case, time point $\top$ is used to represent unbounded persistence. An effect φ of action finishing at t is represented by a fact token $\langle t, \top, \varphi \rangle$.

A time map with a set $\mathcal{F}$ of fact tokens supports queries about whether it guarantees that a fact φ holds over an interval $[t_1, t_2]$ (written as tm_holds($\langle t_1, t_2, \varphi \rangle, \mathcal{F}$)). A fact φ is guaranteed to hold over an interval $[t_1, t_2]$ exactly if there is an interval $[t_1', t_2']$ such that $(t_1' \leq t_1 \leq t_2 \leq t_2') \wedge \langle t_1', t_2', \varphi \rangle \in \mathcal{F}$ and if there is no $\langle t_3, t_4, \varphi' \rangle \in \mathcal{F}$ such that contradicts$(\varphi, \varphi') \wedge (t_1' < t_3 \leq t_2)$.

A precondition ψ of an action can be represented by a triple similar to a fact token. Since it must be satisfied at the action's starting time t, it is represented by $\langle t, t, \psi \rangle$.

[1]This paper uses Shoham's terminology (1994).

start	⟨an integer⟩
end	⟨an integer⟩
action	⟨a term⟩
rsubactions	⟨a sequence of terms⟩
constraints	⟨a set of constraints⟩
effects	⟨a set of triples⟩
preconditions	⟨a set of triples⟩
aend	⟨a variable⟩

Figure 3: *Edge* structure

Data Structures

In our chart parsing, an action is represented by an edge. Since information on the effects and preconditions of the action represented by an edge must be kept locally, we use the *edge* structure shown in Figure 3. An edge's *start* and *end* values are vertices that are the respective integers representing the starting and ending time points of (the part of) the action represented by the edge. The *action* and *rsubactions* (remaining subactions) values are respectively an action description and a sequence of descriptions of actions to find in order to recognize the action. An edge is called active if its *rsubactions* value is a non-empty sequence and is inactive otherwise. The *constraints* value is a set of constraints on variable instantiation. The *effects* and *preconditions* values respectively are sets of triples representing the action's effects and preconditions. The *aend* (action end) value is a variable used as the placeholder of the action's ending time point. The ending time of the action represented by an active edge is not determined yet, and neither is the starting point of the effects specified by the action's recipe. To keep information on those effects in the edge, fact tokens with the *aend* value as their starting time points are used. An unbound time point variable is taken to be greater than any integer and to be less than $\top$. An edge's *aend* value is bound to its *end* value if it is inactive. Given an edge e and its field *field*, *field*(e) denotes the value of *field* in e.

Chart Procedures

Given an observed action, chart parsing applies the following procedure:

Procedure 1 Let α_j be the description of the j-th observed action. For each recipe with action α_r, and for each most general unifier θ of α_j and α_r satisfying the constraints C_r specified by the recipe, create an inactive edge from $j-1$ to j such that its *action*, *constraints*, *effects*, and *preconditions* values respectively are $\alpha_j\theta$, $C_r\theta$, $\{\langle j, \top, \varphi_r\theta \rangle | \varphi_r \in E_r\}$, and $\{\langle j-1, j-1, \psi_r\theta \rangle | \psi_r \in P_r\}$, where E_r and P_r are the effects and preconditions specified by the recipe.

Chart parsing proceeds using the following two procedures.

Procedure 2 Let e_i be an inactive edge. For each recipe with its action α_r, decomposition $\langle\gamma_1,\ldots,\gamma_n\rangle$, effects E_r, and preconditions P_r, and for each most general unifier θ, satisfying *constraints*(e_i) and recipe's constrains C_r, of *action*(e_i) and γ_1 such that

$$\begin{aligned}\mathcal{E} &= (\mathit{effects}(e_i))\theta \\ &\quad \cup\{\langle v,\top,\varphi_r\theta\rangle \mid \varphi_r \in E_r\} \text{ and} \\ \mathcal{P} &= \{\langle t,t,\psi\rangle \in (\mathit{preconditions}(e_i))\theta \mid \\ &\qquad \neg\text{tm\_holds}(\langle t,t,\psi\rangle,\mathcal{E})\} \\ &\quad \cup\{\langle \mathit{start}(e_i),\mathit{start}(e_i),\psi_r\theta\rangle \mid \psi_r \in P_r\},\end{aligned}$$

without E-P or P-P conflict, where v is a new variable, create an edge from *start*(e_i) to *end*(e_i) such that its *action*, *rsubactions*, *constraints*, *effects*, *preconditions*, and *aend* values respectively are $\alpha_r\theta$, $\langle\gamma_2,\ldots,\gamma_n\rangle\theta$, $(C_r \cup \mathit{constraints}(e_i))\theta$, $\mathcal{E}$, $\mathcal{P}$, and v.

Procedure 3 Let e_a and e_i be adjacent active and inactive edges such that *rsubactions*(e_a) is $\langle\gamma_1,\ldots,\gamma_n\rangle$. For each most general unifier θ, satisfying $C = \mathit{constraints}(e_a) \cup \mathit{constraints}(e_i)$, of γ_1 and *action*(e_i) such that

$$\begin{aligned}\mathcal{E} &= (\mathit{effects}(e_a) \cup \mathit{effects}(e_i))\theta \text{ and} \\ \mathcal{P} &= \{\langle t,t,\psi\rangle \in (\mathit{preconditions}(e_a) \\ &\qquad \cup \mathit{preconditions}(e_i))\theta \mid \\ &\qquad \neg\text{tm\_holds}(\langle t,t,\psi\rangle,\mathcal{E})\},\end{aligned}$$

without E-P or P-P conflict, create an edge from *start*(e_a) to *end*(e_i) such that its *action*, *rsubactions*, *constraints*, *effects*, *preconditions*, and *aend* values respectively are $(\mathit{action}(e_a))\theta$, $\langle\gamma_2,\ldots,\gamma_n\rangle\theta$, $C\theta$, $\mathcal{E}$, $\mathcal{P}$, and *aend*(e_a).

Now that we have the basic means to calculate the effects and preconditions of the action represented by an edge, we can augment plan parsing to handle the situations described in Section 2.

Effect-based action descriptions The fact that the description of the form achieve(P) can specify an action with P as its effect is captured by augmenting Procedures 2 and 3. The set of effects of the action represented by an inactive edge e_i that hold at the action's ending time is $E_i = \{\varphi \mid \text{tm\_holds}(\langle\varphi, \mathit{end}(e_i), \mathit{end}(e_i)\rangle, \mathit{effects}(e_i))\}$. The fact is thus captured in these procedures by checking that E_i contains P, instead of unifying γ_1 with *action*(e_i), if γ_1 is of that form.

The fact that achieve(P) can specify the null action if P already holds is captured by a new procedure that, given an active edge e_a with as its *rsubactions* value $\langle\text{achieve(P)},\gamma_2,\ldots,\gamma_n\rangle$, creates a new edge whose *rsubactions* value is $\langle\gamma_2,\ldots,\gamma_n\rangle$ and whose *preconditions* value is *preconditions*(e_a) if e_a has P as its effect and *preconditions*(e_a) plus $\langle \mathit{end}(e_a), \mathit{end}(e_a), \text{P}\rangle$ otherwise.

Action-enabling An action-enabling relationship can be captured by a new procedure that, given two adjacent inactive edges e_1 and e_2 such that e_1's effects satisfy some of e_2's preconditions, creates a new inactive edge with *action*(e_2) as its *action* value.[2]

State-dependent interpretation A dialogue state is determined by the initial state and the effects of the preceding actions. The initial state is treated by using a special 'initialize' inactive edge from 0 to 0 with the *effects* value representing it. The influence of the 'initialize' edge is propagated by the procedure for treating action-enabling relationships and preference rules referring to preconditions.[3]

5 Conclusion

A plan parsing method has been proposed that handles the effects and preconditions of actions and that parses plans in a manner dependent on dialogue state changes caused by utterances. The method has been implemented in Prolog. The implemented program uses an agenda mechanism that uses priority scores on edges to obtain preferred plans first. The method has been applied to understanding route-explanation dialogues by using the dialogue plan model that takes each action of uttering a word as a primitive and that treats intra- and inter-utterance plans uniformly to treat fragmentary utterances (Kogure et al., 1994).

References

Thomas L. Dean and Drew V. McDermott. 1987. Temporal data base management. *Artificial Intelligence*, 32(1):1–55.

Kiyoshi Kogure, Akira Shimazu, and Mikio Nakano. 1994. Recognizing plans in more natural dialogue utterances. In *Proceedings of ICSLP 94*, pages 935–938.

Akira Shimazu, Kiyoshi Kogure, and Mikio Nakano. 1994. Cooperative distributed processing for understanding dialogue utterances. In *Proceedings of ICSLP 94*, pages 99–102.

Yoav Shoham. 1994. *Artificial Intelligence Techniques in Prolog*. Morgan Kaufmann Publishers.

Candace L. Sidner. 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10.

Mark Vilain. 1990. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of AAAI-90*, pages 190–197.

[2] As an extention to control the applicability of this procedure, the *effects* and *preconditions* fields respectively are divided into *main_effects* and *side_effects* fields and into *preconditions* and *prerequisites* fields. The procedure checks enabling relationships only between *main_effects* and *preconditions*.

[3] The use of the initial state also contributes to the efficiency of plan parsing: an input action sequence can be shortened by calculating the current state in the middle of a dialogue and by restarting plan parsing with the current state as a new initial state.