Recording Rationale in <I-N-C-A> for Plan Analysis

Wickler, G., Potter, S., & Tate, A. (2006). Recording Rationale in <I-N-C-A> for Plan Analysis. In Workshop on Plan Analysis and Management, International Conference on Automated Planning and Scheduling (ICAPS-06).

Recording Rationale in <I-N-C-A> for Plan Analysis

Gerhard Wickler, Stephen Potter, Austin Tate

Artificial Intelligence Applications Institute
University of Edinburgh, Edinburgh, Scotland
{g.wickler, s.potter, a.tate}@ed.ac.uk

Abstract

The aim of this paper is to show how the rationale behind a plan can be recorded in the plan itself. The <I-N-C-A> model which underlies the I-X framework will be described in detail, focussing on annotations. It is there that a planner can record the justifications for including components into the plan. Recording rationale information of this type can be used for a number of purposes in the life cycle of a plan, including plan indexing and retrieval, failure recovery, plan explanation and establishing trust as explained in this paper.

Introduction

Plans are the artefact that is the result of the planning process. Traditionally, a plan is described as a set of activities together with some organizational structure, e.g. a sequence in the simplest case (Ghallab *et al*., 2004). This is a simplistic model of a plan that can only be applied in toy domains where plans cannot go wrong, need not be stored for later re-use, need not be justified, etc. This view of plans ignores a lot of the knowledge that is generated and used during the planning process. In this paper we will describe the <I-N-C-A> model of a plan (Tate, 2003) which can store annotations to record knowledge about the plan that is generated during the planning process. Specifically, we want to record the rationale behind some (but not all) planning decisions in the plan itself for later use (e.g. during plan execution, re-planning or explanation generation). This knowledge can be used to facilitate plan analysis and help maintain the plan as a meaningful entity.

Background

Rationale has been recognized as an important type of information in the planning literature. In fact it can be traced back to early work on Hacker's plan teleology (Sussman, 1973), Nonlin's "Goal Structure" (Tate, 1977; Tate, 1983) and work on Plan Rationale in SIPE (Wilkins, 1988). Plan rationale capture and use is a key research objective in the I-X framework (Tate, 2000; Potter *et al*., 2003; Wickler *et al*., 2006) and its predecessor O-Plan (Currie and Tate, 1991; Tate *et al*., 2000).

One of the fundamental ideas here is that it is necessary to have a clear and sharable ontology of plans before one can reason about plans (Tate, 1996; 1998). The <I-N-C-A> model of a plan represents such an ontology and will be described in detail in this paper, focusing on the component that is used to record the rationale behind the plan.

Rationale is an essential component of knowledge-rich plans (Polyak and Tate, 1998). Having such models not only facilitates the planning process itself, but also makes it possible to analyse and re-use such plans. Plan rationale can be viewed in terms of causality, dependencies and decisions. Each of these dimensions addresses practical issues in the planning process and adds value to the resultant plan.

The <I-N-C-A> Model in I-X

<I-N-C-A> is a generic model for synthesis tasks (Tate, 2003). While its level of abstraction makes it possible to apply the generic model to a wide variety of tasks, it assumes a more specific meaning in the I-X agent framework when the object to be synthesized is a plan, a course of action the I-X agent intends to follow.

Terminology

In this section we will introduce some of the terminology used in the description of <I-N-C-A> that follows. This is necessary as we use the terms explained here with specific meanings.

World-State Propositions. We assume here that a state of the world can be described by a set of world-state propositions. By a world-state proposition we mean any logical expression that represents a proposition about the world that can be true or false, and not necessarily a proposition in propositional logic. <I-N-C-A> does not commit to any specific formalism for world-state propositions. Traditionally world-state propositions are described as first-order literals or state-variable expressions in AI planning, but more complex formalisms may be required to reason about, for example, the knowledge of agents in a world state.

Primitive and Complex Activities. Primitive activities are considered to be the atomic elements that make up the plan. They are primitive in the sense that, from the perspective of the planner, they can be executed directly.

A primitive activity must be an instantiation of some activity schema defined in the planning domain. An activity schema contains variables representing the parameters necessary to describe fully the activity: For primitive activities to be executable these parameters must

have specific values. The name of each activity schema must be unique within a planning domain, whereas there can be multiple primitive activities with the same activity name in a plan. In classical planning primitive activities are often called actions (Ghallab *et al*., 2004).

Complex activities are not primitive in that, from the perspective of the planner, they cannot be executed directly but instead need to be refined or broken down into primitive activities that can be executed. In Hierarchical Task Network (HTN) planning complex activities are often called tasks (Ghallab *et al*., 2004) or processes (Tate, 1998). Together, primitive and complex activities constitute the set of all activities. Note that this terminology applies at the object-level, i.e., referring to entities in the domain in question, as well as at the meta-level relating to the planning process itself, as described below.

Note too that the choice of which activities are primitive and which complex depends on the context and knowledge of the agent in question: usually an activity will be modelled as primitive if it can be carried out in one step from this perspective, and as complex otherwise.

Plans. An instantiation of the <I-N-C-A> model is an <I-N-C-A> object. In the I-X framework an <I-N-C-A> object is synonymous with a plan. A plan can be partial in the sense that it is not (yet) an actionable solution to a planning problem. It is the job of the planner to refine a partial plan into a solution plan.

The <I-N-C-A> Representation in I-X

Planning can be described as synthesizing an <I-N-C-A> object, i.e., a plan, in which *nodes* are activities. We can formally define an <I-N-C-A> object in I-X as a 4-tuple (*I,N,C,A*) consisting of:

- a set of *issues I*,
- a set of *activity nodes N*,
- a set of *constraints C*, and
- a set of *annotations A*.

Issues. *I* is the set of unresolved *issues* in the current plan, i.e., in this <I-N-C-A> object. An issue is represented by a syntactic expression of the form $l{:}M(O_1,\ldots,O_n)$, where:

- l is a unique label for this issue,
- M is a symbol denoting a primitive plan modification activity, and
- $O_1,\ldots,O_n$ are plan-space objects, i.e. they are issues, nodes, constraints or annotations. The number of such objects, n, and the interpretation of each object in the context of the issue, will depend on the particular primitive plan modification activity represented by this issue.

Issues can be seen as primitive meta-level activities, i.e. things that need to be done to the plan before it becomes a solution to a given planning problem. This approach is inherited from O-Plan (Currie and Tate, 1991; Tate *et al*., 2000) and is also seen in planners such as OPIS (Smith, 1994). The most commonly found primitive meta-level activities carried out by planners, but usually only implicit in their underlying implementation or internal plan representation, are:

- Achieving a goal (in classical planners): Let p be a world-state proposition and τ be a time point, then the primitive meta-level activity of achieving p at τ can be represented as the issue:

$$l_1\texttt{:achieve(}p\texttt{,}\tau\texttt{)}$$

- Accomplishing a complex activity (in HTN planners): Let $a \in N$ be a complex activity. Then the primitive meta-level activity of accomplishing a can be represented as the issue:

$$l_2\texttt{:refine(}a\texttt{)}$$

Here, `achieve` and `refine` are examples of symbols denoting primitive plan modification activities. Note that these symbols are not domain specific but specific to the planning process by which these types of issue are handled. Issues can be either 'negative', in which case they can be thought of as flaws in the plan, or they can be 'positive', e.g., opportunities.

An alternative view of issues now being explored in recent I-X research is to see them as always expressed as questions that need to be answered. For example, the primitive meta-level activity of refining a can be phrased as the question "How can a be accomplished?" Adopting this view, issues can then be classified and manipulated according to the question types (Conklin, 2005) described in recent advances based on the large body of work on issue-based design (Conklin and Begeman, 1988).

An <I-N-C-A> object is considered to be a solution to a planning problem only if the set of issues is empty.

Nodes. N is the set of activities (*nodes*) to be performed in the current plan, i.e., in this <I-N-C-A> object. An activity is a syntactic expression of the form $l{:}\alpha(o_1,\ldots,o_n)$, where:

- l is a unique label for this activity,
- α is a symbol denoting an activity name, and
- $o_1,\ldots,o_n$ are object-level terms, i.e. they are either constant symbols describing objects in the domain, or they are as yet uninstantiated variables standing for such objects.

Time points constitute a special class of domain objects that are found as parameters of an activity. Specifically, two time points, one representing the begin and the other the end of an activity, are often used as parameters.

In the context of I-X, nodes represent the object-level activities in the plan, i.e., things that need to be performed by some agent to execute the plan. As mentioned above, activities can be of two types from the perspective of the planner:

- Primitive activities: primitive activities can be carried out directly by an agent executing the plan. For example, in a search and rescue domain, the primitive activity of flying the aircraft `ac1` from location `loc1` to location `loc2` may be represented as:

$$l_3\texttt{:fly(ac1,loc1,loc2)}$$

- Complex activities: complex activities cannot be accomplished directly by the agent executing the plan but need to be refined into primitive activities. For example, the complex activity of rescuing an isolated person `ip` may be represented as:

$$l_4\texttt{:rescue(ip)}$$

In this example, `fly` is a primitive activity symbol and `rescue` is a complex activity symbol in some domain. Activity symbols have to be domain specific. It follows that there has to be an activity schema defined for the domain that has the name `fly` and describes when this activity schema is applicable and how it will change the world when applied, and there has to be a refinement defined in the domain that accomplishes a complex activity with the name `rescue` and describes how exactly it can be accomplished.

Note that the set N of activities in the plan may contain both complex activities and the primitive activities that have been chosen to implement them.

Constraints. C is the set of *constraints* that must be satisfied by the current plan (<I-N-C-A> object). A constraint is a syntactic expression of the form $l{:}c(v_1,\ldots,v_n)$, where:

- l is a unique label for this constraint,
- c is a symbol denoting a constraint relation, and
- $v_1,\ldots,v_n$ are constraint variables, i.e., they can represent domain objects (including time points), variables in activities (which may have binding constraints attached).

Constraints represent the relations that must hold between the different objects related in the constraints for the plan to be executable. In the context of planning, the most commonly used constraints are of the following types:

- Ordering constraints: Let v_1, v_2 be variables in the plan representing time points. Then the constraint that v_1 has to be before v_2 can be represented as:

$$l_5\texttt{:before(}v_1\texttt{,}v_2\texttt{)}$$

- World-state constraints: Let p be a world-state proposition and v a variable representing a time point in the plan. Then the fact that p is a condition that has to hold at the time point represented by v, or the fact that p is an effect of an activity that holds at time point v can be represented respectively as:

$$l_6\texttt{:cond(}p\texttt{,}v\texttt{)}$$

$$l_7\texttt{:effect(}p\texttt{,}v\texttt{)}$$

- Variable binding constraints: Let v be a variable mentioned in some activity $a\in N$ and s be a constant symbol in the planning domain. Then the fact that v must take the value s can be represented as:

$$l_8\texttt{:value(}v\texttt{,}s\texttt{)}$$

These are just some of the constraint types that can be defined. The objects related to each other can be of different types. This is reflected by the domains of the constraint variables representing them. They can be world-state propositions as in conditions and effects, or they can be variables used in activities representing time points or other domain objects in the plan as in ordering and variable binding constraints.

Annotations. A is the set of *annotations* attached to the current plan. Amongst other things, annotations can be used to add human-centric information to the plan. They may be informal or they may adhere to some detailed syntax (which is not specified as part of <I-N-C-A>).

Annotations can be used to record arbitrary information about the plan (and the annotations form a part of this plan – hence the plan becomes, in some sense, self-descriptive). Specifically, in this paper we want to discuss the annotation of plans with one particular type of rationale, namely the rationale information that can be recorded by the planner during the planning process. In this case, an annotation will be a syntactic expression of the form $l_a{:}r\,(l_p{:}O\texttt{,}\,l_m{:}M\texttt{,}\,O_1,\ldots,O_n)$, where:

- l_a: is a unique label for this annotation,
- r is a rationale predicate relating a plan-space object to other plan-space objects,
- $l_p{:}O$ is a labelled plan-space object that is part of the current plan, i.e., it is an issue, an activity, a constraint or an annotation,
- $l_m{:}M$ is an issue that was formerly in the plan and has since been resolved, i.e., it is a primitive meta-level activity that has been performed by the planner, and
- $O_1,\ldots,O_n$ are plan-space objects that may or may not be labelled.

An annotation of this type represents the fact that the plan-space object O was introduced into the plan as part of performing the plan modification activity M, and possibly involving other plan-space objects $O_1,\ldots,O_n$. The rationale predicate r denotes the relationship between these objects and describes the justification for including O. Thus, the interpretation of such an annotation depends on the rationale predicate r used. The different labels are necessary to specify the exact object that is being referred to. This is necessary as there might be two activities in the plan which are identical except for the label. The following examples illustrate the use of rationale annotations of this form.

- Let $l_m{:}\texttt{achieve(}p\texttt{,}\tau\texttt{)}$ be an issue in the current plan and let $\alpha(o_1,\ldots,o_n)$ be an activity schema defined in the domain that has an effect that unifies with p under the substitution σ. Suppose the planner introduces a new activity $l_p{:}\sigma(\alpha(o_1,\ldots,o_n))$ into the plan to address the issue $l_m{:}\texttt{achieve(}p\texttt{,}\tau\texttt{)}$. Then the following annotation can be added to the plan to record the rationale for adding $l_p{:}\sigma(\alpha(o_1,\ldots,o_n))$:

$$\texttt{naap(}l_p{:}\sigma(\alpha(o_1,\ldots,o_n))\texttt{,}\,l_m{:}\texttt{achieve(}p\texttt{,}\tau\texttt{)}\texttt{,}\,p\texttt{)}$$

In this case `naap` is a rationale predicate that expresses that a new activity, the first argument, was introduced into the plan to address the issue of achieving some proposition (the second and third arguments respectively). Thus, the argument types for this particular rationale predicate are an activity $a\in N$, an issue $m\in I$ in which the plan modification activity symbol is

`achieve`, and a world-state proposition. Furthermore, the last argument, the proposition *p*, must be the same as the one to be achieved in the plan modification activity, and it must be unifiable with one of the effects of the activity $a \in N$.

In this case, a second rationale annotation could be introduced in a similar fashion to express the fact that l_p:σ($\alpha(o_1,\ldots,o_n)$ has to be performed before the time point τ.

- Let l_m:`refine(`*a*`)` be an issue in the current plan and let there be a refinement Δ defined in the domain that can be used to accomplish *a* under the substitution σ by refining it into, amongst other things, activities $\sigma(\alpha_1(o_1,\ldots,o_n))\ldots\sigma(\alpha_k(o_1,\ldots,o_n))$. Note that the elements into which *a* is refined can together be seen as an <I-N-C-A> object, i.e. they can be issues, nodes, constraints and annotations. Suppose the planner uses Δ to refine *a* and this adds new activities l_{p1}:$\sigma(\alpha_1(o_1,\ldots,o_n))\ldots l_{pk}$:$\sigma(\alpha_k(o_1,\ldots,o_n))$ to *N* to address the issue l_m:`refine(`*a*`)`. Then, the following annotation can be added to the plan to record the rationale for adding each l_{pi}:$\sigma(\alpha_i(o_1,\ldots,o_n))$, $1 \le i \le k$:

  `nadi(`l_{pi}:σ($\alpha_i(o_1,\ldots,o_n)$)`,` l_m:`refine(`*a*`)`, Δ`)`

  (One such annotation would be added for each new activity α_i.) In this case `nadi` is a rationale predicate that expresses that a new activity, the first argument, was introduced into the plan to address the issue of refining some proposition in accordance with some particular refinement in the domain (the second and third arguments respectively). Thus, the argument types for this rationale predicate must be an activity $a \in N$, an issue $m \in I$, where the plan modification activity symbol has to be `refine`, and a refinement. Furthermore, the last argument, the refinement Δ, must be defined as accomplishing a complex activity that can be unified with *a*.

  Similarly, if appropriate, analogous rationale annotations could be introduced to express the fact that other <I-N-C-A> elements of the refinement – such as issues or constraints – were also introduced as part of this refinement.

Rationale predicates of this type are usually specific to a type of issue. Hence, `naap` rationale will always relate to an `achieve` issue, and `nadi` rationale will always relate to a `refine` issue. However, there may be multiple rationale predicates that may be used with the same issue – that used will depend on how the planner did actually resolve the issue. For example, achieving a proposition at some time point can be done by introducing a new activity before the time point or by maintaining the truth of the proposition if it was true at another, previous time point. Thus, the relation between rationale predicates and issues is not one-to-one: issues need not always be resolved in the same manner.

Note too that this type of rationale, recording justifications for the inclusion of objects into the plan, is only one type of rationale that one may want to record in a plan. For example, we may want to record why a specific way of refining a plan was chosen among the various available options. While we believe that this type of information would be very useful to record, we believe that this will best be approached by use of a separate decision structure. It is in general not possible to extract useful knowledge of this kind from a search-based planning algorithm that tries out many possibilities and backtracks upon failure. At any choice point, there may be a large number of reasons why all the leaf nodes that are in the search space under the choice point represent failures in the search, and it may be hard to abstract these into meaningful rationale. However, there also exist choice points in a search space where a decision is forced or made via user selectyion from open alternatives and it may be most useful to record this as part of the rationale for the plan. This is not described here though.

Issues as Questions

In the I-X framework, until recently, issues had a task or activity orientation to them, being mostly concerned with actionable items referring to the process underway – i.e., actions in the process space. This is now not felt to be appropriate, and we are adopting the gIBIS (Conklin and Begeman, 1988) orientation of expressing these issues as any of a number of specific types of question to be considered (Selvin, 1999; Conklin, 2005). The types of questions advocated are:

1. Deontic questions – What should we do?
2. Instrumental questions – How should we do it?
3. Criterial questions – What are the criteria?
4. Meaning or conceptual questions – What does *X* mean?
5. Factual questions – What is *X*? or Is *X* true?
6. Background questions – What is the background to this project?
7. Stakeholder questions – Who are the stakeholders of this project?
8. Miscellaneous questions – To act as a catch all.

The first 5 of these are likely to be the most common in our task support environment. This is similar to the Questions - Options - Criteria approach (MacLean *et al*., 1991) - itself used for rationale capture for plans and plan schema libraries in our earlier work (Polyak and Tate, 1998; 2000) and similar to the concept mapping approaches used in Compendium (Selvin et al. 2001). Compendium can in fact exchange its set of issues, activities and some types of constraints and annotations with I-X (Buckingham Shum *et al*., 2002; Chen-Burger and Tate, 2003).

The Uses of Rationale

Fundamental to the <I-N-C-A> model is the idea of maintaining annotations as first-class elements placed alongside the more conventional elements of a plan. One of the principal uses of annotations is to capture rationale; hence, we consider rationale to be an important element of

this model, and rationale capture and expression are areas which we are currently exploring.

The approach outlined in the previous sections, , should be seen as a framework and tentative steps towards defining a typology of plan rationale and corresponding mechanisms for its capture. These tasks are necessarily guided by the uses to which we want to put this rationale; hence, in this section we discuss briefly some of the types of operations and reasoning that we hope to support through the capture of rationale. In general terms, these are intended to support activity in *real* domains (as opposed to classical planning domains and puzzles). In other words, domains in which we accept that information and knowledge may be imprecise, incorrect or missing, and as a result, we expect plans to fail – and expect that the use of rationale will enable us to *fail better*.

Explanation and Trust

As might be expected, a major use of rationale is for explaining the existence of particular elements in the plan, e.g., why a certain activity (rather than any other) appears in the plan. This becomes particularly important when trying to decide if the plan can be re-applied in the current context, or if execution of the plan fails or partially fails (of which, more later). Another use of explanation, one particularly important in mixed-initiative (i.e., human and computer) agent systems arises when we wish to *justify* a certain activity, particularly in those cases where we are asking another agent to perform this activity. In all but the most rigidly enforced hierarchical systems, where agents simply obey commands (and which occur very rarely in practice), we should expect that any agent might respond to such a request with a request of its own demanding that the activity be justified (and that, if the activity cannot be justified to the agent's satisfaction, it might refuse to perform the activity). It should be apparent that rationale would allow us to supply some justification. Moreover, through the use of <I-N-C-A> objects as our common interlingua in the domain, this justification can be included and communicated as part of the activity. In this way, the object may be thought of as analogous to the idea of proof-carrying code, in that the presence of the rationale can help convince the recipient of the appropriateness of performing the activity and that it is 'safe' to be performed in the current situation.

This sort of transaction and reasoning can be seen as an important step for establishing *trust* between agents. Notions of trust, and ways in which it can be established and managed, are currently receiving much attention among those considering open agent architectures, particularly Semantic Web and Semantic Web Services researchers, where it is considered to be vital if these initiatives are to come to full fruition.

Plan Indexing and Retrieval

Often re-use of existing plans will be more appealing than planning anew for a particular task. One use of rationale is for richer indexing (and later retrieval) of plans; alongside the description of what the plan does (expressed in the through the plan itself), and the constraints under which it is applicable. The rationale annotations allow us to access the reasons why the plan does what it purports to do. Properly captured, this information would allow us to avoid plan re-use under inappropriate conditions or avoid choosing plans that are based on (what are now known to be faulty) assumptions or judgements, or at least to be aware of these limitations and deal with them accordingly.

Failure Recovery and Replanning

In the real world it is inevitable that some plans will fail; even the best-laid plans can be undermined by some unexpected event. The failure may or may not be important with respect to the plan rationale. We need to separate unimportant minor side-effects from failures which impact on the intended results of the plan (Tate, 1984; Reece and Tate, 1994; Drabble and Tate, 1997). In such cases, it is very likely that we will need to do something to recover, and to do this efficiently, we will need to try to understand why the plan has failed, and hence, when replanning to help guide the choice of alternative actions that may overcome this failure.

Explanation-Based Plan Learning

Since the plan is accompanied by some explanation of why it is considered valid (in the form of the rationale), this suggests the possibility of learning about the domain from both positive and negative examples (plan successes and failures). This learning may help to, for instance, identify and repair faulty knowledge or assumptions, and provide modified rational criteria for choice of particular options over others.

Conclusions

In this paper we have presented an approach to recording the rationale behind a plan in the plan itself, thus making the plan a self-contained entity that does not require knowledge of the planning algorithm to explain the structure of the plan. Fundamental to this approach is the <I-N-C-A> model which can be used to describe synthesis tasks and has been used in the I-X framework for synthesizing plans. Issues in this model can be described as meta-level activities that are performed by the planner to refine the plan. During this planning process the planner adds new constraints on the space of possible behaviour to the plan, and each of these constraints is added for a reason. It is this type of rationale that we can record as annotations in <I-N-C-A> in order to be able to better understand the plan, the result of the planning process. This knowledge-rich plan can then be used in various ways outlined in this paper, thus facilitating the use of the plan in a wider context.

Acknowledgments

The I-X project is sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number F30602-03-2-0014. Parts of this work are supported by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC) sponsored by the UK Engineering and Physical Sciences Research Council by grant no. GR/N15764/01. The University of Edinburgh and research sponsors are authorized to reproduce and distribute reprints and on-line copies for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of other parties.

References

Buckingham Shum, S., De Roure, D., Eisenstadt, M., Shadbolt, N. and Tate, A. (2002) CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid, *Proceedings of the Second Workshop on Advanced Collaborative Environments*, Eleventh IEEE Int. Symposium on High Performance Distributed Computing (HPDC-11), July 24-26, 2002, Edinburgh, Scotland.

Chen-Burger, Y. and Tate, A. (2003) Concept Mapping Between Compendium and I-X, Informatics Report Series, University of Edinburgh, EDI-INF-RR-0166, May 2003.

Conklin J. and Begeman M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 4(6), pp 303-331.

Conklin J. (2005) Dialogue Mapping: Building Shared Understanding of Wicked Problems, Wiley.

Currie K. and Tate A.. (1991) O-Plan: the Open Planning Architecture. *Artificial Intelligence*, Vol. 52, pp 49-86.

Drabble, B. and Tate, A. (1997) Repairing Plans on the Fly, in *Proceedings of the NASA Workshop on Planning and Scheduling for Space*, Oxnard CA, USA, October 1997

Ghallab M., Nau D., and Traverso P. (2004) *Automated Planning – Theory and Practice*,. Elsevier/Morgan Kaufmann.

MacLean A., Young R., Bellotti V. and Moran T. (1991) Design space analysis: Bridging from theory to practice via design rationale. In *Proceedings of Esprit* '91, pages 720-730, Brussels, November 1991.

Polyak S. and Tate A. (1998) Rationale in Planning: Causality, Dependencies and Decisions, *Knowledge Engineering Review*, Vol.13 (3), pp. 247-262, September, 1998, Cambridge University Press. See http://www.aiai.ed.ac.uk/project/oplan/documents/1998/98-rationale.pdf

Polyak S. and Tate A. (2000) A Common Process Ontology for Process-Centred Organisations, Knowledge based Systems, 2000. Earlier version by S. Polyak published as University of Edinburgh Department of Artificial Intelligence Research paper 930, 1998. See http://www.aiai.ed.ac.uk/project/oplan/documents/1999/99-sebpc-cpm.pdf

Potter, S., Tate, A. and Dalton, J. (2003) I-X: Task Support on the Semantic Web, Poster Abstract, Second International Semantic Web Conference (ISWC-2003), Sanibel Island, Florida, October 2003.

Reece, G. and Tate, A. (1994) Synthesizing Protection Monitors from Causal Structure, in *Proceedings of the Second International Conference on Planning Systems (AIPS-94)*, Chicago, June 1994, AAAI Press.

Selvin A.M. (1999) Supporting Collaborative Analysis and Design with Hypertext Functionality, Journal of Digital Information, Volume 1 Issue 4.

Selvin, A.M, Buckingham Shum, S.J., Sierhuis, M., Conklin, J., Zimmermann, B., Palus, C., Drath, W., Horth, D., Domingue, J., Motta, E. and Li, G. (2001) Compendium: Making Meetings into Knowledge Events, *Knowledge Technologies* 2001, Austin TX, USA, March 4-7, 2001.

Smith, S.F. (1994) OPIS: A methodology and architecture for reactive scheduling. In Zweben, M. and Fox, M.S. (eds), *Intelligent Scheduling*, chapter 8, pages 29--66. Morgan Kaufmann, San Francisco, CA, USA.

Sussman, G. J. (1973) "A Computational Model of Skill Acquisition," *Technical Report 297*, MIT AI Lab, Cambridge, MA.

Tate, A., (1975) "Generating Project Networks", *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)* pp. 888-893, Boston, Mass. USA, August 1977. Reprinted in *Readings in Planning,* Morgan-Kaufmann, 1990.

Tate A. (1983) The Less Obvious Side of Nonlin. Department of Artificial Intelligence, University of Edinburgh. http://www.aiai.ed.ac.uk/project/oplan/documents/1990-PRE/1983-unpub-tate-nonlin-less-obvious.pdf

Tate, A. (1984) Planning and Condition Monitoring in a FMS, *Proceedings of the International Conference on Flexible Automation Systems*, Institute of Electrical Engineers, London, UK, July 1984.

Tate A.. (1996) Towards a Plan Ontology. *AI*IA Notiziqe* (Quarterly Publication of the Associazione Italiana per l'Intelligenza Artificiale), Special Issue on "Aspects of Planning Research", Vol. 9. No. 1, pp.19-26 - March 1996

Tate A.. (1998) Roots of SPAR - Shared Planning and Activity Representation, *The Knowledge Engineering Review*, Vol 13(1), pp. 121-128, Special Issue on Putting Ontologies to Use (eds. Uschold. M. and Tate, A.), Cambridge University Press, March 1998.

Tate, A. (2000) Intelligible AI Planning, in Research and Development in Intelligent Systems XVII, *Proceedings of*

ES2000, The Twentieth British Computer Society Special Group on Expert Systems International Conference on Knowledge Based Systems and Applied Artificial Intelligence, pp. 3-16, Cambridge, UK, December 2000, Springer.

Tate A.., Dalton J. and Levine J. (2000) O-Plan: a Web-based AI Planning Agent, AAAI-2000 Intelligent Systems Demonstrator, in *Proceedings of the National Conference of the American Association of Artificial Intelligence* (AAAI-2000), Austin, Texas, USA, August 2000.

Tate A.. (2003) An Ontology for Mixed-initiative Synthesis Tasks. *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems (MIIS)* at the International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico, August 2003, pp 125-130.

Wickler G., Tate A., and Potter S. (2006) Using the <I-N-C-A> Artifact Model as a Shared Representation of Intentions for Emergency Response Coordination. In: Jennings N., Tambe M., Ishida T., Ramchurn G. *Proc. First International Workshop on Agent Technology for Disaster Management*, Japan, May 2006.

Wilkins, D.E. (1988) Practical Planning: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, San Mateo, CA, USA.