Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015

Deliverable Number:	D3.2.1
Deliverable Title :	Structural Bootstrapping for planning: informed search for knowledge-level planning
Type (Internal, Restricted, Public):	PU
Authors:	Ron Petrick
Contributing Partners:	UEDIN

Contractual Date of Delivery to the EC:	31-01-2012
Actual Date of Delivery to the EC:	31-01-2012

Contents

Executive Summary

A central contribution of WP3 (Generative Mechanisms), and WP3.2 (Structural Bootstrapping for Planning) in particular, is to extend the capabilities of current high-level planning models by applying structural bootstrapping to the knowledge-rich representation of actions and plans, and to provide the apparatus needed to support plan generation and execution in low-level robotics domains (WP2; Outside In: Development and Representations) and higher-level domains requiring language and communication (WP4; Interaction and Communication). To this end, we describe the work by UEDIN on knowledge-level planning during the last work period (M1–M13), with a focus on Task 3.2.1 (Informed Search Methods) and Task 3.2.4 (Extended Reasoning about Object and Indexical Knowledge). This work is also related to the project-wide integration and demonstrations of WP5 (System Integration).

The ability to reason and plan is essential for an intelligent agent acting in a dynamic and incompletely known world, such as the proposed robot scenarios in WP5. High-level planning capabilities in Xperience are (partly) supplied by the PKS planner [8, 9], which UEDIN is extending for use in robotic and linguistic domains. PKS is a state-of-the-art conditional planner that constructs plans in the presence of incomplete information. Unlike traditional planners, PKS builds plans at the "knowledge level", by representing and reasoning about how the planner's knowledge state changes during plan generation. Actions are specified in a STRIPS-like [4] manner in terms of action preconditions (state properties that must be true before an action can be executed) and action effects (the changes the action makes to properties of the state). PKS can build conditional plans with sensing actions, and supports numerical reasoning, run-time variables [3], and features like functions that arise in real-world planning scenarios.

Like most AI planners, PKS operates best in discrete, symbolic state spaces described using logical languages. As a result, work that addresses the problem of integrating planning on real-world robot platforms often centres around the problem of representation, and how to abstract the capabilities of a robot and its working environment so that it can be put in a suitable form for use by a goal-directed planner. Integration also requires the ability to communicate information between system components. Thus, the design of a planning system often has to take into consideration external concerns, to ensure proper interoperability with modules that aren't traditionally considered in pure theoretical planning settings. Furthermore, a planner must strive for efficiency, to overcome the computation challenges arising from operating in real-world environments like the proposed robot scenarios of Xperience.

In order to address the efficiency concerns of planning in complex domains, as part of Task 3.2.1 we have begun to investigate *informed search methods* from the wider planning community that could be adapted to the PKS planning system. Most modern planning systems rely on a search process to find a plan that achieves the goal of a planning problem. However, searching through large state spaces of the kind that typically arise in real-world robot domains is often problematic, since the search may take a large amount of time, if it completes at all, and requires significant resource overhead. Thus, most modern planners employ some form of informed or heuristic search in an attempt to gain leverage on the structure of a given search problem, in order to improve the efficiency of the plan generation process.

In addition to exploring the efficiency problem, we have also made significant contributions in extending PKS's representation language as part of Task 3.2.4, by providing a method for reasoning with numerical information that can be used to model the effects of noisy sensors and noisy effectors at the planning level. Since such scenarios are commonplace in real-world robot domains, we believe these extensions will be particularly useful in integrating high-level planning with the low-level sensorimotor systems on Xperience.

The two papers attached to this deliverable, [Pet12] and [Pet11], provide details of our contributions in the past work period to the above tasks.

Overall, this deliverable reports a number of significant developments:

- We have completed initial studies into the task of adapting *informed search techniques* to the PKS planner. A strategy has been adopted to explore both *problem relaxation* methods, which seek to explore simpler versions of a state space at the search level, and *compilation* methods, which attempt to transform the original problem specification into a simpler form that can be more easily solved. Both methods make use of rich domain knowledge and structure which is a central focus of structural bootstrapping on Xperience.
- Initial extensions to PKS's *plan generation and search* mechanisms in support of future heuristic search methods have been completed.
- A theoretical model of *reasoning with numerical information* in PKS has been developed, based on the concept of an interval-valued function. This model includes an algorithm for progressing knowledge states with a representation of noisy sensors and noisy effectors. Such a representation provides a middle ground between planners that cannot work with numerical properties and those that use full probabilistic representations. A prototype version of PKS with these extensions has been implemented.
- In conjunction with KIT, we have begun developing *high-level domain descriptions* for use with PKS on the ARMAR platform. Such a domain will model some of the sophisticated capabilities of the ARMAR robot, including object manipulation with multiple grippers and movement between workspaces in its operating environment.
- A new version of a *plan execution monitor* for PKS is currently being developed with an early prototype ready for testing. This module works in conjunction with the backend PKS planner to provide plan monitoring and replanning services, and can communicate with other system modules using a modern ICE middleware interface.

A number of tasks remain open at the time of this report and constitute ongoing and future work:

- The plan execution monitor currently being developed has not been tested on the robot hardware.
- We are continuing to investigate the role of probabilistic and numeric models in high-level plan generation and monitoring processes, with our work in [Pet11] being our first contribution here. Since nondeterminacy will undoubtedly arise as the result of perception and action at the sensorimotor level, we are studying how best to utilise such information at the higher control levels. Currently, we are experimenting with is the use of rapid replanning [10] (in conjunction with our plan execution monitor) which has been successfully applied by planners in the probabilistic track of the International Planning Competition [5].
- We have focused on the planner-level concerns of representation and efficiency in this report, however, this workpackage will also explore an approach that applies modern planning techniques to problems in natural language generation (e.g., [7, 1, 6, 2]). In particular, we will generalise our existing apparatus for ordinary action planning to dialogue planning with speech acts. Initial theoretical work is currently underway, ahead of the planned schedule.

In addition to the general workpackage connections mentioned above, we have also had specific interactions with partners from KIT and SDU through email and Skype calls, and face-to-face meetings with all project partners at an Xperience working meeting hosted by UEDIN.

Bibliography

[1] Luciana Benotti. Accommodation through tacit sensing. In *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 75–82, London, United Kingdom, 2008.

[2] Michael Brenner and Ivana Kruijff-Korbayová. A continual multiagent planning approach to situated dialogue. In *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 67–74, 2008.

[3] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In William Swartout, Bernhard Nebel, and Charles Rich, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 115–125, Cambridge, MA, October 1992. Morgan Kaufmann Publishers.

[4] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[5] ICAPS. International Planning Competition. *http://ipc.icaps-conference.org*, 2008.

[6] Alexander Koller and Ronald Petrick. Experiences with planning for natural language generation. In *Scheduling and Planning Applications woRKshop (SPARK 2008) at ICAPS 2008*, September 2008.

[7] Alexander Koller and Matthew Stone. Sentence generation as planning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 336–343, Prague, Czech Republic, 2007.

[8] Ronald P. A. Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, pages 212–221, Menlo Park, CA, April 2002. AAAI Press.

[9] Ronald P. A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 2–11, Menlo Park, CA, June 2004. AAAI Press.

[10] Sungwook Yoon, Alan Fern, and Robert Givan. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-07)*, pages 352–359, 2007.

Attached Papers

[Pet12] Ronald P. A. Petrick. Knowledge-level planning in the Xperience project. Xperience Technical Report, University of Edinburgh, January 2012.

Abstract: This report describes part of UEDIN's contribution to the ongoing work of high-level planning in the Xperience project, namely knowledge-level planning. The focus of this document is a description of the role of the PKS (Planning with Knowledge and Sensing) planner, and its derivative technologies, which will be used and extended throughout the project. This document will continually be updated for each reporting period and provide a status report of key aspects of the PKS-related research agenda on Xperience. In the first reporting period of this work (M13), we present an overview of the problem of informed search for knowledge-level planning, and describe a workplan for extending the PKS planner (or its successor) during the next work period of the project. We also highlight other research directions for knowledge-level planning currently in progress.

[Pet11] Ronald P. A. Petrick. An extension of knowledge-level planning to interval-valued functions. In *AAAI 2011 Workshop on Generalized Planning*, San Francisco, CA, USA, August 2011.

Abstract: We investigate the problem of reasoning about numerical functions in the presence of incomplete information, sensing actions, and conditional plans. An interval-based representation is introduced into the PKS (Planning with Knowledge and Sensing) planner, as a means of compactly representing sets of possible values for numerical functions. We describe the enhancements we make to PKS, and demonstrate how such information can be used for modelling uncertain sensors and effectors. We also show how interval-valued functions can be used as a form of noisy run-time variable in plans. This paper presents a snapshot of work currently in progress.

KNOWLEDGE-LEVEL PLANNING IN THE XPERIENCE PROJECT

Ron Petrick
University of Edinburgh
rpetrick@inf.ed.ac.uk

2012-01-25

Abstract

This report describes part of UEDIN's contribution to the ongoing work of high-level planning in the Xperience project, namely knowledge-level planning. The focus of this document is a description of the role of the PKS (Planning with Knowledge and Sensing) planner, and its derivative technologies, which will be used and extended throughout the project. This document will continually be updated for each reporting period and provide a status report of key aspects of the PKS-related research agenda on Xperience. In the first reporting period of this work (M13), we present an overview of the problem of informed search for knowledge-level planning, and describe a workplan for extending the PKS planner (or its successor) during the next work period of the project. We also highlight other research directions for knowledge-level planning currently in progress.

Revision history

2012-01-25 : Initial report focusing on an overview of the problem of informed search for knowledge-level planning and strategies for implementing such a search process into PKS.

Contents

1 Introduction

In this document we describe the state of knowledge-level planning work by UEDIN in the Xperience project. This work forms part of WP3 (Generative Mechanisms) and, in particular, WP3.2 (Structural Bootstrapping for Planning). The present report primarily addresses Task 3.2.1 (Informed Search Methods) and presents the results of initial exploratory studies into informed search methods and how they could be adapted to a knowledge-level planning. This work is also closely related to WP4 (Interaction and Communication) and the project-wide integration work and demonstrations of WP5 (System Integration).

High-level planning capabilities in the Xperience project are (partly) supplied by the PKS planner [Petrick and Bacchus, 2002, 2004], which UEDIN is extending for use in robotic and linguistic domains as part of WP3 (with some connections to WP4). PKS is a state-of-the-art knowledge-level planner that constructs plans in the presence of incomplete information. Unlike traditional planners, PKS builds plans at the "knowledge level", by representing and reasoning about how the planner's knowledge state changes during plan generation. Actions are specified in a STRIPS-like [Fikes and Nilsson, 1971] manner in terms of action preconditions (state properties that must be true before an action can be executed) and action effects (the changes the action makes to properties of the state). PKS is able to construct conditional plans with sensing actions, and supports numerical reasoning, run-time variables [Etzioni et al., 1992], and features like functions that arise in real-world planning scenarios.

Like most AI planners, PKS operates best in discrete, symbolic state spaces described using logical languages. As a result, work that addresses the problem of integrating planning on real-world robot platforms often centres around the problem of representation, and how to abstract the capabilities of a robot and its working environment so that it can be put in a suitable form for use by a goal-directed planner. Integration also requires the ability to communicate information between system components. Thus, the design of a planning system often has to take into consideration external concerns, to ensure proper interoperability with modules that aren't traditionally considered in pure theoretical planning settings.

PKS was successfully used on the PACO-PLUS project, a predecessor project of Xperience. On that project, integration work successfully established a link between PKS and KIT's ARMAR robot platform. In Xperience, we will build on these past successes to extend this integration even further. In particular, since we will address more complex challenges in the robot domains we will consider, the high-level planning capabilities must be extended on a number of levels. First, representations must be extended to improve our ability to model real-world problems at the planning level. This will particularly be important for the dialogue planning work scheduled for a later work period of the project. However, changes to the representation alone are not enough. The plan generation methods themselves must be improved to take advantage of the new features of an extended representation language. Furthermore, as the complexity of the planning problem increases, we must work harder to find methods to overcome the difficulties arising from increased domain size leading to increased planning times. Methods such as informed search control seek to address these concerns by improving the efficiency of the plan generation process itself.

In the remainder of this document we will present a brief overview of the state of knowledge-level planning in Xperience. In Section 2, we address the problem of informed search and present our workplan for achieving some of our defined goals in the next work period. In Section 3 we highlight the state of some of the other research directions we are currently exploring for knowledge-level planning in Xperience.

2 Knowledge-level planning and informed search

2.1 Background to planning and informed search

The ability to reason and plan is essential for an intelligent agent acting in a dynamic and incompletely known world—such as the proposed robot scenarios in Xperience. Achieving goals under such conditions often requires complex forward deliberation that cannot easily be achieved by simply reacting to a situation without considering the long term consequences of a course of action.

The problem of *planning* has been extensively studied in artificial intelligence. One of the most influential approaches has been STRIPS [Fikes and Nilsson, 1971], a representation language that reduces the problem of specifying actions to that of describing an action's preconditions (i.e., the qualification problem) apart from an action's effects. A solution to the frame problem [McCarthy and Hayes, 1969] is also captured in STRIPS as a persistence condition on properties unaffected by an action. Although STRIPS traditionally makes certain unrealistic assumptions about the nature of a planning domain, namely that actions are deterministic and world states are completely known, it nevertheless forms the core of PDDL [McDermott et al., 1998], the standard language of many modern planners and the language of the International Planning Competition [ICAPS, 2008]. The success of this representation has led to the development of many modern STRIPS-based planners that use techniques like heuristic search [Hoffmann and Nebel, 2001] or compilation [Palacios and Geffner, 2007] to scale to large problem instances.

While pure STRIPS-based planners are not directly relevant to Xperience, due to their representational limitations, alternate attempts to solve more general planning problems often use variants of STRIPS as their underlying representations (e.g., [Cimatti and Roveri, 2000, Bonet and Geffner, 2001]). In Xperience, we use an approach based on *knowledge-level planning*, an instance of the general problem of *planning with incomplete information and sensing* (e.g., [Peot and Smith, 1992])—i.e., planning with incomplete states and observational actions that return information about the world state. In contrast to more traditional approaches which build plans by reasoning about how actions affect the state of the world, the knowledge-level approach builds plans at a more abstract level, by directly representing and reasoning about the incompleteness of the planner's knowledge and how that knowledge changes during planning. By abstracting the type of reasoning used during plan generation, this approach has the potential to generate quite complex plans very efficiently. This approach also has links to the *knowledge representation and reasoning* community and those logical accounts that restrict epistemic expressivity for tractable reasoning (e.g., [Funge, 1998, Demolombe and Pozos Parra, 2000, Son and Baral, 2001, Liu and Levesque, 2005, Vassos and Levesque, 2007]), as an alternative to more traditional accounts of knowledge and action (e.g., [Moore, 1985, Scherl and Levesque, 2003]). A similar idea to this is also explored in the EU FP7 CogX project (ICT-215181).

One of the most advanced knowledge-level planners is PKS ("Planning with Knowledge and Sensing") [Petrick and Bacchus, 2002, 2004], a contingent planner that constructs plans with incomplete information and sensing. Unlike most planners, PKS uses an extended STRIPS representation but restricts the types of knowledge it can represent in exchange for more efficient reasoning. PKS is particularly adept at modelling knowledge-level changes resulting from sensing actions, which arise in many challenging planning scenarios. PKS also supports many features needed for real-world planning, such as the representation of functional information, numerical reasoning, and run-time variables [Etzioni et al., 1992]. PKS has been successfully used in complex physical environments such as the robot kitchen domain in the FP6 PACO-PLUS project, and is being used to control a robot bartender in the FP7 JAMES project (Grant No. 270435) that must interact with multiple human users.

Reasoning about sensing actions allows planners to control certain types of indefinite information that arise in the world. However, related approaches to *planning under uncertainty* also attempt to manage other types of nondeterminism, including actions with noisy effects or probabilistic outcomes. Currently, the most successful techniques employ rapid replanning methods that make use of advances in heuristic search [Hoffmann and Nebel, 2001], a technique we will extend to knowledge-level planning in Xperience. Planners like FF-Replan [Yoon et al., 2007] have been successfully employed in domains such as those in the probabilistic track of the International Planning Competition [ICAPS, 2008].

In Xperience, planning will also be used for the purpose of natural language dialogue, and the demands of that problem will influence the structure and requirements of a suitable planning system. The tasks of natural language generation and reasoning about dialogue have long traditions of using planning approaches. Early approaches to generation as planning (e.g., [Perrault and Allen, 1980, Appelt, 1985, Hovy, 1988, Young and Moore, 1994]) focused primarily on high-level structures, such as speech acts and discourse relations, but suffered due to the inefficiency of the planners available at the time. As a result, recent mainstream research has tended to segregate task planning from discourse and dialogue planning, capturing the latter with more specialised approaches such as finite state machines, information state approaches, speech-act theories, dialogue games, or theories of textual coherence [Lambert and Carberry, 1991, Traum and Allen, 1992, Green and Carberry, 1994, Young and Moore, 1994, Chu-Carroll and Carberry, 1995, Matheson et al., 2000, Beun, 2001, Asher and Lascarides, 2003, Maudet, 2004].

There has also been a renewed interest in applying modern planning techniques to problems in NLG, such as sentence planning [Koller and Stone, 2007], instruction giving [Koller and Petrick, 2008], and accommodation [Benotti, 2008]. The idea of viewing interaction management as a planning problem has also being revisited, for instance by identifying the problem of planning conversational moves as an instance of the general problem of planning with incomplete information and sensing actions [Stone, 2000]. This view is also implicit in early "beliefs, desires and intentions" (BDI)-based approaches, e.g., [Litman and Allen, 1987, Bratman et al., 1988, Cohen and Levesque, 1990, Grosz and Sidner, 1990]. Thus, certain types of communicative actions (e.g., speech acts like "asking" and "telling") are treated as ordinary sensing actions that return otherwise unknown information to the agent. Initial work using the knowledge-level PKS planner explored this connection [Steedman and Petrick, 2007], but fell short of implementing a robust tool that could leverage this relationship for efficient dialogue planning. A related approach from the FP6 CoSy project [Brenner and Kruijff-Korbayová, 2008] also attempted to manage dialogues by interleaving planning and execution, but failed to solve the consequent problem of deciding when best to commit to plan execution versus plan construction. Thus, many planning approaches are promising, but not yet fully mature, and fall outside the mainstream of most recent NLG and dialogue work.

The common thread in all these approaches is that they seek to overcome the computational challenges inherent in complex domains. The most successful approach arising from the planning community has been the use of heuristic or informed search techniques for this task.

2.2 Informed search strategies

Most modern planning systems rely on the process of *search* to find a sequence of actions that transforms an initial state into a state satisfying the goal of the planning domain. However, a search through a large state space of the kind that typically arises in many robot domains is problematic: the search process may take a large amount of time to complete, if it completes at all, due to the resource overhead required to explore large state spaces. As a result, most modern planners employ some form of *informed*

search or *heuristic search* (either directly or indirectly) in an attempt to gain leverage on the structure of a given search problem, in order to improve plan generation times.

Surveying the recent literature on informed search in the planning community shows that there are two main strategies employed by existing planning systems:

1. **Problem relaxation**: In problem relaxation, rather than solving the original problem in the original search space, the problem space is abstracted in some way, typically by considering a subset or generalisation of the original planning space. The relaxed problem, which is typically a simpler problem, is solved and then used as an estimate for guiding the search in the original problem. In this case, solving the relaxed problem must be done in an efficient manner so that the planner has time to apply its results to the original problem with a net gain in overall planning time.
2. **Compilation methods**: The idea of compilation usually refers to the process whereby the original problem domain is transformed into an alternate, usually simpler problem instance, and an existing (potentially more efficient) tool is then used to solve the simpler problem. The solution to the compiled problem is then applied to the original problem, sometimes through a process that modifies the generated plan in some predefined way. Such methods may lead to exact compilations where a problem can be provably reduced to a simpler problem instance, or approximate compilations where the reduced problem is not provably "exact", but nevertheless produces a solution that can be used to solve the original problem.

(A third approach also exists, which is namely a hybrid approach that combines the two approaches. For instance, as part of the problem relaxation approach a compilation method might be employed to simplify the existing problem first before relaxing it at the search space level.)

Examples of both approaches exist in the planning literature and have been successful in practice. The most famous planner based on problem relaxation is FF [Hoffmann and Nebel, 2001] which uses a relaxation technique that ignores the "delete lists" in a STRIPS action description (i.e., the properties of the world that become false when an action is applied). In this case, FF generates over-specified states during its search which provides a lower bound on the number of steps a plan requires before a domain property could possibly become true. This estimate, which is generated as part of a structure called a planning graph, is then used to inform the original search problem. In many of the standard planning benchmarks, such as those from the International Planning Competition, this heuristic has been shown to be quite effective and had lead to impressive performance on these domains. The success of this approach has also influenced a number of subsequent planners which have attempted to adapt the FF approach for improved performance. The task of designing new relaxation heuristics for search remains an active area of research within the planning community.

The most influential work on compilation techniques within the planning community is that of Palacios and Geffner [2007], which investigate the problem of converting a conformant planning problem (a problem with incomplete information but no sensing actions) into a classical problem (a problem with complete information) that can then be solved using an off-the-shelf planner like FF. This is done by converting the original problem (described in terms of world-level properties) to its knowledge level counterpart, by introducing new fluents into the domain model. When viewing the problem at the knowledge level, a closed-world view of the fluents can be used, allowing a planner like FF to be employed. Once a solution to the compiled problem is found, it is translated back to the original problem, in some cases by expanding or removing certain actions in the process. Compilation approaches are promising because they allow existing tools to be leveraged to solve subproblems. However, usually the compilation imposes restrictions on the types of problems that such techniques can be used on. For those domains for

which compilation techniques do not apply, approximate solutions might be possible, or else other techniques must be adopted.

2.3 Informed search strategies for PKS

There are two immediate research questions for the knowledge-level planning component on Xperience concerning informed search: (1) which technique should we use, and (2) how should it be adapted to a planner like PKS? First, we observe that neither technique offers an immediate solution that can simply be "lifted" into a planner like PKS. First, the state spaces that PKS uses are different from those of planners like FF, and the relaxation technique of ignoring delete lists does not immediately transfer. Second, the representation language used by PKS is substantially more expressive than those used in the planners that are considered by Palacios and Geffner. As a result, the actual compilation technique used in Palacios and Geffner [2007], and other similar work, does not immediately translate to PKS.

However, the situation is not all bad news. First, while the FF relaxation technique does not directly apply to PKS, relaxation techniques that are applicable should be possible. The challenge in this case is identifying an approach that solves an abstraction of the PKS planning problem quickly and accurately, and that can be used to inform a heuristic search process. This remains a challenging but essential area of research for the knowledge-level planning agenda at UEDIN.

Second, while the technique of [Palacios and Geffner, 2007] also doesn't immediately translate to PKS, a comparable approach used in earlier work, which does consider a subset of the PKS representation, is applicable [Petrick and Levesque, 2002, Petrick, 2006]. This approach uses ideas similar to the work Palacios and Geffner but is based on the logical language of the situation calculus. As a result, this approach provides a much needed logical theory, which is less clear in [Palacios and Geffner, 2007].

As a result, we are adopting a two-pronged strategy for PKS. First, we will explore relaxation techniques that can be used directly with the state spaces that arise in the PKS planner. This approach will consist of a substantial implementation stage (which is currently underway) to restructure the codebase of the PKS planner which implements search, to take advantage of informed heuristic estimates. The code restructuring will also provide a testbed whereby the PKS planning algorithm can be used with various informed search techniques, to evaluate which are more effective at certain types of domains arising on the Xperience project.

Second, we are also extending the work of [Petrick and Levesque, 2002, Petrick, 2006] to provide a theoretical understanding of how compilation can be used in PKS. Part of this work will out of necessity revisit the work of [Palacios and Geffner, 2007] and re-interpret it in terms of the work of Petrick and Levesque. We believe that many of the techniques presented in [Petrick and Levesque, 2002] can be adapted to a more practical planning setting, but also that some of the insights of [Palacios and Geffner, 2007] have a role in improving the existing theoretical models in the situation calculus. We also believe that the revised compilation technique can then be used in conjunction with a PKS-specific relaxation method, combining the two approaches. However, more research is needed to investigate what form such a technique might take.

2.4 Proposed timeline

We provide the following timeline covering M1–M24 for the informed search task. We note that this timeline is subject to change and the work that is necessary for completing this task may extend beyond the end of the next work period.

Month	Task	Status
	Studies	
M1–M12	Initial informed search studies	Complete
M12–M16	Intermediate studies	Underway
	Theory development	
M12–M18	Initial heuristic search strategies	Underway
M14–M18	Initial compilation techniques	Not yet started
M18–M24	Intermediate search and compilation techniques	Not yet started
	Implementation and evaluation	
M11–M18	Extension of existing search framework	Underway
M18–M24	Implementation of initial/intermediate search techniques	Not yet started
M11–M24	Evaluation of extended search methods	Ongoing

2.5 Example: compiling open world PKS to closed world FF

As an example of our early thinking on compilation techniques, we apply the approach of Petrick [2006] to compile a simple PKS planning domain into a closed world form that is then solvable using the off-the-shelf FF planner. An example is given using a standard planning benchmark problem, the Bomb-in-the-Toilet domain.

Original PKS Bomb-in-the-Toilet domain

Here we give an encoding of the original Bomb-in-the-Toilet domain in PKS syntax.

Action	Precondition	Effects
dunk(p,t)	$K($*package*$(p))$ $K($*toilet*$(t))$ $K(\neg$*clogged*$(t))$	*add*$(K_f,$ *disarmed*$(p))$ *add*$(K_f,$ *clogged*$(t))$
flush(t)	$K($*toilet*$(t))$	*add*$(K_f, \neg$*clogged*$(t))$

Compiled Bomb-in-the-Toilet domain

We now show the encoding of the compiled Bomb-in-the-Toilet domain using the techniques of [Petrick, 2006]. Note that for each fluent P in the original PKS domain, a pair of *knowledge fluents*, KP and $K\neg p$ are introduced into the compiled version of the problem. In this case, the set of knowledge fluents forms a type of "closed world" problem which can be solved using classical planning techniques.

Action	Precondition	Effects
dunk(p,t)	K*package*(p) K*toilet*(t) $K\neg$*clogged*(t)	$\neg K\neg$*disarmed*(p) $\neg K\neg$*clogged*(t) K*disarmed*(p) K*clogged*(t)
flush(t)	K*toilet*(t)	$\neg K$*clogged*(t) $K\neg$*clogged*(t)

Run times

Finally, we give some early results on solving the compiled version of the PKS domain using the FF planner. In this case the running time is significantly faster using the optimised FF approach, based on heuristic search.

	Compiled to FF		Original PKS	
Problem (#P, #T)	#Actions	Time (s)	#Actions	Time (s)
bomb-50-50	50	0.19	100	1.220
bomb-60-60	60	0.43	120	2.190
bomb-70-70	70	0.82	140	3.640
bomb-80-80	80	1.45	160	5.840
bomb-90-90	90	2.41	180	8.280
bomb-100-100	100	3.83	200	11.78
bomb-100-10	190	0.48	200	3.730
bomb-100-20	180	0.73	200	4.430
bomb-100-30	170	0.91	200	5.180
bomb-100-40	160	1.08	200	5.970
bomb-100-50	150	1.28	200	6.790
bomb-100-60	140	1.56	200	7.730
bomb-100-70	130	1.93	200	8.650

3 Other research directions for knowledge-level planning

In this section we briefly note other research directions for knowledge-level planning currently being investigated. Many of these tasks are at a very early stage and are not planned to be fully active until a later stage of the project.

- **Extended representation for planning with intervals**: This work seeks to improve the planner's ability to work with numerical properties. An interval-valued representation is introduced to model the range of possible mappings for a numeric function. See [Petrick, 2011] for more details.
- **Modelling multiagent knowledge**: This work seeks to extend the planner's representation and reasoning framework to support multiagent knowledge of the form that can be used to model dialogue actions or information returned from external data sources. This work involves both the development of a theoretical model (most likely in the situation calculus) and an extension of the planner's codebase.
- **Plan execution monitoring**: Although much of the work of this workpackage is focused on plan generation, a second high-level component is needed to monitor plan execution and control replanning/resensing activities. This monitor is responsible for assessing both action failure and unexpected state information that result from feedback provided to the planner from the execution of planned actions at the robot level. The difference between predicted and actual states is used to decide between (i) continuing the execution of a plan, (ii) resensing activities that target a portion of a scene at a higher resolution to produce a more detailed state report, and (iii) replanning from new/unexpected states. Building on the initial version of the plan execution monitor developed for the PACO-PLUS project, we are currently extending the monitor to use the new ICE middleware framework being developed for the ARMAR robot at KIT.
- **Domains**: An ongoing task is the design of new planning domains supporting the project's demonstration scenarios. This task will continue throughout the project.

References

D. Appelt. *Planning English Sentences*. Cambridge University Press, Cambridge, England, 1985.

N. Asher and A. Lascarides. *Logics of Conversation*. Cambridge University Press, 2003.

L. Benotti. Accommodation through tacit sensing. In *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 75–82, London, United Kingdom, 2008.

R.-J. Beun. On the generation of coherent dialogue. *Pragmatics and Cognition*, 9: 37–68, 2001.

B. Bonet and H. Geffner. GPT: A tool for planning with uncertainty and partial information. In *Proceedings of the IJCAI-01 Workshop on Planning with Uncertainty and Incomplete Information*, pages 82–87, Aug. 2001.

M. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

M. Brenner and I. Kruijff-Korbayová. A continual multiagent planning approach to situated dialogue. In *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL 2008)*, pages 67–74, 2008.

J. Chu-Carroll and S. Carberry. Response generation in collaborative negotiation. In *Proceedings of ACL-95*, pages 136–143. ACL, 1995.

A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

P. Cohen and H. Levesque. Rational interaction as the basis for communication. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, MA, 1990.

R. Demolombe and M. P. Pozos Parra. A simple and tractable extension of situation calculus to epistemic logic. In *Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems (ISMIS-2000)*, pages 515–524, Oct. 2000.

O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In W. Swartout, B. Nebel, and C. Rich, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 115–125, Cambridge, MA, Oct. 1992. Morgan Kaufmann Publishers.

R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

J. Funge. Interval-valued epistemic fluents. In *AAAI Fall Symposium on Cognitive Robotics*, pages 23–25, Orlando, FL, Oct. 1998.

N. Green and S. Carberry. A hybrid reasoning model for indirect answers. In *Proceedings of ACL-94*, pages 58–65. ACL, 1994.

B. Grosz and C. Sidner. Plans for discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA, 1990.

J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

E. Hovy. *Generating natural language under pragmatic constraints*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1988.

ICAPS. International Planning Competition. *http://ipc.icaps-conference.org*, 2008.

A. Koller and R. Petrick. Experiences with planning for natural language generation. In *Scheduling and Planning Applications woRKshop (SPARK 2008) at ICAPS 2008*, Sept. 2008.

A. Koller and M. Stone. Sentence generation as planning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 336–343, Prague, Czech Republic, 2007.

L. Lambert and S. Carberry. A tripartite plan-based model of dialogue. In *Proceedings of ACL-91*, pages 47–54. ACL, 1991.

D. Litman and J. Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

Y. Liu and H. J. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proceedings of the Ninteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 522–527, 2005.

C. Matheson, M. Poesio, and D. Traum. Modeling grounding and discourse obligations using update rules. In *Proceedings of NAACL 2000*, Seattle, WA, USA, 2000.

N. Maudet. Negotiating language games. *Autonomous Agents and Multi-Agent Systems*, 7:229–233, 2004.

J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language (Version 1.2). Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Oct. 1998.

R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1985.

H. Palacios and H. Geffner. From conformant into classical planning: Efficient translations that may be complete too. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-07)*, pages 264–271, Sept. 2007.

M. A. Peot and D. E. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS-92)*, pages 189–197, College Park, MD, June 1992. Morgan Kaufmann Publishers.

C. R. Perrault and J. F. Allen. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3–4):167–182, 1980.

R. P. A. Petrick. *A Knowledge-level approach for effective acting, sensing, and planning*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 2006.

R. P. A. Petrick. An extension of knowledge-level planning to interval-valued functions. In *AAAI 2011 Workshop on Generalized Planning*, San Francisco, CA, USA, Aug. 2011.

R. P. A. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In M. Ghallab, J. Hertzberg, and P. Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, pages 212–221, Menlo Park, CA, Apr. 2002. AAAI Press.

R. P. A. Petrick and F. Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In S. Zilberstein, J. Koehler, and S. Koenig,

editors, *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 2–11, Menlo Park, CA, June 2004. AAAI Press.

R. P. A. Petrick and H. Levesque. Knowledge equivalence in combined action theories. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-2002)*, pages 303–314, San Francisco, CA, Apr. 2002. Morgan Kaufmann Publishers.

R. B. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.

T. C. Son and C. Baral. Formalizing sensing actions – a transition function based approach. *Artificial Intelligence*, 125(1–2):19–91, 2001.

M. Steedman and R. P. A. Petrick. Planning dialog actions. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue (SIGdial 2007)*, pages 265–272, Antwerp, Belgium, Sept. 2007.

M. Stone. Towards a computational account of knowledge, action and inference in instructions. *Journal of Language and Computation*, 1:231–246, 2000.

D. Traum and J. Allen. A speech acts approach to grounding in conversation. In *Proceedings of ICSLP-92*, pages 137–140, 1992.

S. Vassos and H. Levesque. Progression of situation calculus action theories with incomplete information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2029–2034, 2007.

S. Yoon, A. Fern, and R. Givan. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-07)*, pages 352–359, 2007.

R. M. Young and J. D. Moore. DPOCL: a principled approach to discourse planning. In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 13–20, Kennebunkport, Maine, USA, 1994.

An Extension of Knowledge-Level Planning to Interval-Valued Functions

Ronald P. A. Petrick
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, Scotland, UK
rpetrick@inf.ed.ac.uk

Abstract

We investigate the problem of reasoning about numerical functions in the presence of incomplete information, sensing actions, and conditional plans. An interval-based representation is introduced into the PKS (Planning with Knowledge and Sensing) planner, as a means of compactly representing sets of possible values for numerical functions. We describe the enhancements we make to PKS, and demonstrate how such information can be used for modelling uncertain sensors and effectors. We also show how interval-valued functions can be used as a form of noisy run-time variable in plans. This paper presents a snapshot of work currently in progress.

Introduction and Motivation

An agent operating in a real-world domain often needs to do so with *incomplete information* about the state of the world. An agent with the ability to *sense* the world can also gather additional information to generate plans with *contingencies*, allowing the agent to reason about the possible outcomes of sensed information at plan time, thereby extending its ability to successfully construct plans in uncertain domains.

One particularly useful type of sensed information is *numerical* information. The ability to reason about numbers is often required in many real-world planning contexts, in order to construct plans that work with numeric state properties (e.g., the robot is 10 metres from the wall), manage limited resources (e.g., ensure the robot has enough fuel to complete the task), satisfy numeric constraints (e.g., only grasp an object if its radius is less than 10 cm), or apply arithmetic operations (e.g., advancing the robot forward one step reduces its distance from the wall by 1 metre). The importance of numerical reasoning in planning has previously been recognized with the inclusion of numeric state variables in PDDL (Fox and Long 2003), and the construction of planning systems like MetricFF (Hoffmann 2003) that are able to work with (limited forms of) numeric information.

Reasoning about numerical information under conditions of incomplete information is potentially problematic, however, especially for planners that are built on possible-world representations or sets of belief states. In such representations, the complete set of possible values for an unknown (or incompletely known) state property is often explicitly represented, e.g., by a set of states, each of which denotes a possible configuration of the actual world state. If the value of a *numeric function* is unknown, then the belief state must contain a state representing every possible mapping of the function, which could be a potentially large (or even infinite) set. Even when the range of possible values is relatively small, the number of required states can quickly grow. E.g., if a numeric function f could potentially map to any natural number between 1 and 100, then we require 100 states to capture the set of possible mappings using a possible world/belief state approach. The state explosion resulting from large sets of mappings can be computationally difficult for planners that must reason with individual states and progress (or regress) those states to construct plans.

The general problem of reasoning about knowledge and action, while avoiding the drawbacks of possible worlds, has previously been studied in formal representation languages like the situation calculus (see, e.g., (Demolombe and Pozos Parra 2000; Soutchanski 2001; Liu and Levesque 2005; Petrick 2006; Vassos and Levesque 2007)). Many of these accounts model certain (restricted) types of knowledge directly, rather than indirectly inferring such information from sets of worlds, thereby trading representational expressiveness for more tractable reasoning. (For instance, simple relational facts can be explicitly modelled by sets of predicates known to be true and sets of predicates known to be false.) One representation for modelling uncertain numerical information without possible worlds uses the notion of an *interval-valued function* as a means of capturing a set of disjunctive alternatives (Funge 1998). The idea is simple: rather than representing each possible function mapping individually across a set of worlds, a single function mapping is used and only the endpoints of the range of possible values are represented. Thus, a function f that maps from 1 to 100 can be denoted in an interval-valued form, $f = \langle 1, 100 \rangle$.

Interval-valued models of numeric information have been investigated in the planning community, especially when time is represented as a resource (see, e.g., (Edelkamp 2002; Frank and Jónsson 2003; Laborie 2003)). The idea of bounding uncertain numeric properties by intervals has also been studied in a planning context (Poggioni, Milani, and Baioletti 2003), however, to the best of our knowledge the combination of numerical reasoning with incomplete information, sensing, and contingent planning has not been fully explored. We focus on this problem in the present paper, which describes work currently in progress to extend the

PKS (Planning with Knowledge and Sensing) planner (Petrick and Bacchus 2002; 2004), by incorporating the ability to use interval-valued functions.

Our interest in adding interval-valued representations to PKS is twofold. First, PKS has always had the ability to work with simple numerical information (e.g., function (in)equalities and arithmetic operations), however, unlike other types of knowledge in PKS, its ability to reason about uncertain numerical values is limited. We believe interval-valued functions provide a compact representation that can be used to model the effects of noisy actions and knowledge, and augment PKS's existing ability to work with incomplete knowledge without using possible worlds or belief states. Second, we are also interested in tracking the results of certain types of sensed information through subsequent physical actions. For instance, if the location of a robot is sensed and the robot then moves 2 steps forward, we should be able to use such information in a planning context, even if the actual value of the robot position isn't explicitly known. We believe an interval-based representation will also be useful in modelling such situations. We explore both of these ideas in this paper.

The rest of the paper is organized as follows. In the next section we briefly review the PKS planner. We then describe a simple model of interval-valued functions. Using this model we characterize the changes we have currently implemented in PKS. We then demonstrate the extended version of PKS with a series of detailed examples. Finally, we discuss some open problems and future work, and conclude.

Planning with Knowledge and Sensing (PKS)

In this work, we aim to extend PKS (Planning with Knowledge and Sensing), a contingent planner that constructs plans in the presence of incomplete information and sensing actions (Petrick and Bacchus 2002; 2004). PKS works at the "knowledge-level" by reasoning about how the planner's knowledge state, rather than the world state, changes due to action. PKS works with a restricted subset of a first-order language, and a limited amount of inference in that subset, allowing it to support a rich representation with non-propositional features such as functions and variables. This approach differs from planners that work with propositional representations over which complete reasoning is feasible, or approaches that model incomplete knowledge based on sets of possible worlds. By working at the knowledge level, PKS can often abstract its reasoning from irrelevant distinctions that occur at the world level.

PKS is based on a generalization of STRIPS (Fikes and Nilsson 1971). In STRIPS, the state of the world is modelled by a single database. Actions update this database and, by doing so, update the planner's world model. In PKS, the planner's knowledge state, rather than the world state, is represented by a set of five databases, each of which models a particular type of knowledge. The contents of these databases have a fixed, formal interpretation in a modal logic of knowledge. Actions can modify any of the databases, which has the effect of updating the planner's knowledge state. To ensure efficient inference, PKS restricts the type of knowledge (especially disjunctions) that it can represent in each database. We briefly discuss each database below.

$\mathbf{K_f}$: This database is similar to a standard STRIPS database except that both positive and negative facts are permitted and the closed world assumption is not applied. K_f is used for modelling the effects of actions that change the world. K_f can include any ground literal ℓ, where $\ell \in K_f$ means "the planner knows ℓ." K_f can also contain information about known function (in)equality mappings.

$\mathbf{K_w}$: This database models the plan-time effects of "binary" sensing actions. $\phi \in K_w$ means that at plan time the planner either "knows ϕ or knows $\neg\phi$," and that at execution time this disjunction will be resolved. In such cases we will also say that the planner "knows whether ϕ." Know-whether knowledge is important since PKS can use such information to construct conditional plans with branches (see below).

$\mathbf{K_v}$: This database stores information about function values that will become known at execution time. In particular, K_v can model the plan-time effects of sensing actions that return constants, such as numeric values. K_v can contain any unnested function term f, where $f \in K_v$ means that at plan time the planner "knows the value of f." At execution time the planner will have definite information about f's value. As a result, PKS is able to use K_v terms as "run-time variables" (Etzioni et al. 1992) or placeholders in its plans.

$\mathbf{K_x}$: This database models the planner's "exclusive-or" knowledge of literals, namely that the planner knows "exactly one of a set of literals is true." Entries in K_x have the form $(\ell_1|\ell_2|\ldots|\ell_n)$, where each ℓ_i is a ground literal. Such formulae represent a particular type of disjunctive knowledge that is common in many planning scenarios, namely that "exactly one of the ℓ_i is true."

$\mathbf{LCW}$: This database stores the planner's "local closed world" information (Etzioni, Golden, and Weld 1994), i.e., instances where the planner has complete information about the state of the world. We mention LCW here for completeness but will not focus on it in this paper.

PKS's databases can be inspected through a set of *primitive queries* that ask simple questions about the planner's knowledge state. Primitive queries have the following form: (i) Kp, is p known to be true?, (ii) $K_v t$, is the value of t known?, (iii) $K_w p$, is p known to be true or known to be false? (i.e., does the planner know-whether p?), or (iv) the negation of queries (i)–(iii). An inference algorithm evaluates primitive queries by checking the contents of the databases, taking into consideration the interaction between different types of knowledge.

An action in PKS is modelled by a set of *preconditions* that query the agent's knowledge state, and a set of *effects* that update the state. Action preconditions are simply a list of primitive queries. Action effects are described by a collection of STRIPS-style "add" and "delete" operations that modify the contents of individual databases. For example, $\mathit{add}(K_f, \phi)$ adds ϕ to K_f, and $\mathit{del}(K_w, \phi)$ removes ϕ from K_w. Actions are permitted to have ADL-style context-dependent effects (Pednault 1989), where the secondary preconditions of an effect are described by lists of primitive

queries, and can employ a limited form of quantification ($\forall^K x$ and $\exists^K x$) that ranges over known instantiations of x.

PKS constructs plans by reasoning about actions in a simple forward-chaining manner: if the preconditions of an action are satisfied by the planner's knowledge state, then the action's effects are applied to the state to produce a new knowledge state. For actions with context-dependent effects, secondary preconditions are similarly evaluated against the knowledge state to determine if their effects should be applied. Planning then continues from the resulting state.

PKS can also add conditional branches to a plan, provided it has K_w knowledge. For instance, if $\phi \in K_w$ then PKS can construct two conditional branches in a plan: along one branch (the K^+ branch) ϕ is assumed to be known (i.e., ϕ is added to K_f), while along the other branch (the K^- branch) $\neg\phi$ is assumed to be known (i.e., $\neg\phi$ is added to K_f). Planning continues along each branch from the new knowledge states, until each branch satisfies the *goal*, also specified as a list of primitive queries.

Interval-Valued Functions and Knowledge

In this paper we will only focus on *functions* that map to numerical values (rather than general constants or terms). For instance, $\mathit{robotLoc} = 10$ might denote a function indicating that a robot is known to be 10 metres from a wall.

An *interval-valued function* is a function whose denotation is an *interval* of the form $\langle u, v\rangle$. The values u and v are called the *endpoints* of the interval, and indicate the bounds on a range of possible mappings for the function. Since we are primarily interested in reasoning about an agent's (incomplete) knowledge during planning, a mapping of the form $f = \langle u, v\rangle$ will mean that the value of f is known to be in the interval $\langle u, v\rangle$.[1] For instance, $\mathit{robotLoc} = \langle 5, 10\rangle$ might indicate that the distance to a wall is known to be between 5 and 10 metres. If a function maps to a *point interval* of the form $\langle u, u\rangle$, for some u, then the mapping is certain and known to be equal to u.

Each interval-valued function will be associated with a particular *number system* $\mathbb{X}$ that restricts the range of permissible intervals for a function. Typically, the number system will be one of the standard mathematical number systems (e.g., the reals $\mathbb{R}$, the natural numbers $\mathbb{N}$, the integers $\mathbb{Z}$, etc.), extended to include the points at infinity, ∞ and $-\infty$. Given a number system $\mathbb{X}$, a mapping $f = \langle u, v\rangle$ is permitted, provided $u, v \in \mathbb{X}$ and $u \leq v$. For every number system $\mathbb{X}$, the special interval $\langle \bot, \top\rangle$ represents the *maximal interval* for that number system. For instance, $\langle \bot, \top\rangle \stackrel{\text{def}}{=} \langle -\infty, \infty\rangle$ in $\mathbb{R}$, however in $\mathbb{B}$, the binary number system consisting of the two elements 0 and 1, $\langle \bot, \top\rangle \stackrel{\text{def}}{=} \langle 0, 1\rangle$. In terms of knowledge, a mapping of the form $f = \langle \bot, \top\rangle$ means that the agent considers every element of $\mathbb{X}$ as a possible mapping for f. In other words, the value of f is completely unknown to the agent.

[1] We will only focus on closed intervals whose endpoints are included as possible mappings (i.e., intervals of the form $[u, v]$ in standard mathematical notation). Open intervals (u, v), or partially-open intervals $(u, v]$ and $[u, v)$, are treated in a similar manner except for minor differences in the boundary cases.

For simplicity, we will assume that all interval-valued functions in this paper range over $\mathbb{N}$ unless otherwise indicated. Also, as an alternative to using the maximal interval $\langle \bot, \top\rangle$ to represent functional uncertainty, we will sometimes use PKS's ability to reason about incomplete information when a function is not listed in its knowledge bases.

Representing and Reasoning about Interval-Valued Knowledge in PKS

In this section we describe some of the changes we have made to PKS to support interval-valued functions. Since this paper presents a snapshot of work currently in progress, we will discuss many of these changes at a high level and leave many of the technical details for a future paper. In particular, we will focus on the representation of interval knowledge by considering changes to the K_f, K_v, K_w, and K_x databases. We will also briefly mention extensions to PKS's primitive query language and action representation.

K_f and knowledge of intervals Recall that the K_f database stores the planner's knowledge of facts, including functional equalities (e.g., $f = 10$) and inequalities (e.g., $g \neq 12$). In our extended representation we allow functions to map to interval values, provided the intervals only contain numeric constants. That is, a function like $f = \langle 5, 10\rangle$ is permitted, however, $g = \langle 5, x\rangle$ is not if x is a variable. Intuitively, a function of the form $f = \langle u, v\rangle \in K_f$ means that f is known to map to a value between u and v.

K_v and sensed intervals The K_v database is primarily used to represent the results of sensing actions that return functions. In particular, this database does not constrain the type of underlying function it can represent, i.e., whether it is an ordinary function mapping or an interval-valued mapping. Thus, K_v can immediately be used with interval-valued functions which are treated in the same way as ordinary functions. I.e., if $f \in K_v$, where f is interval valued, then the (interval) value of f is known at plan time.

However, we also extend our notion of K_v knowledge to allow *noisy* sensed information to be modelled. To do so, we specify an *interval schema* for the associated function, using a variable (x in our examples) to denote the actual value of the function. For instance, a function of the form:

$$f : \langle x - 1, x + 1\rangle \in K_v$$

means that the value of the function f is known, and f is in the range $x \pm 1$, for some x. In this case, we treat x as a special type of "run-time variable" (Etzioni et al. 1992) that acts as a placeholder to the actual value of the sensed function. The value of f in this case is "noisy" as it admits a range of possible values. In practice, we allow formulae of the form $f : \langle x - u, x + v\rangle$ in K_v, where u and v are numeric constants. This type of information will be particularly useful for tracking changes to numeric sensed values through the effects of certain physical actions.

K_w and numeric comparisons The K_w database is typically used to model sensing actions with binary outcomes, i.e., those that return one of two possible values. K_w is also important since information in this database can be used to

build conditional branches into a plan: when a conditional branch is inserted, one branch is added for each possible outcome of the sensed information.

With numeric functions (interval-based or not), certain types of numeric relations become useful in a planning context. In particular, the relational operators $=$, $\neq$, $>$, $<$, $\geq$, and $\leq$ often arise in many planning scenarios. In our extended version of PKS, we allow simple formulae using such operators to be explicitly represented in K_w, provided such formulae have the form f **op** c, where **op** $\in \{=, \neq, >, <, \geq, \leq\}$ and c is a numeric constant. Thus, $f > 5 \in K_w$ can be used to model a sensing action that determines whether f is greater than 5 or not.

Such extended K_w information can also be used to form conditional plans as usual in PKS. For a given K_w formula, two branches are added to a plan: along one branch the K_w formula is assumed to be true while along the other branch the formula is assumed to be false. Thus, if the formula $f > 5 \in K_w$ is used as the basis for a new branch point in a plan then $f > 5$ is assumed to be true in the K^+ branch, and $f \leq 5$ is assumed to be true in the K^- branch.

K_w branching is particularly important when combined with interval-based knowledge in K_f: any assumptions resulting from the addition of a branch must be combined with existing knowledge in the other databases, possibly refining or resolving that knowledge as necessary. Thus, if $f > 5$ is assumed to be true and $f = \langle 3, 10\rangle \in K_f$, then the K_f knowledge is updated and the interval is refined so that $f = \langle 6, 10\rangle$. Similarly, if $f \leq 5$ is assumed to be true then the K_f knowledge is updated so that $f = \langle 3, 5\rangle$. When used with interval-based knowledge that has a wide range of possible values, this process gives rise to a powerful technique that allows the planner to split intervals into smaller components and employ a form of case-based reasoning.

K_x versus interval-valued functions The notion of an interval-valued function has a close connection to the exclusive-or knowledge that can be represented in K_x: both types of representation can be used to model disjunctions of possible values where one, and only one, of the disjunctions can be true. For instance, in this view a formula of the form $(f = 3 \mid f = 4 \mid f = 5) \in K_x$ is similar to an interval-valued function of the form $f = \langle 3, 5\rangle$.

There are notable differences, however. In particular, K_x takes a very conservative view of physical actions that change the values of literals mentioned in K_x formula. In such cases, any formula containing a property changed by an action is completely removed from K_x since it's "exclusive-or" property may no longer hold. This is not the case for interval-valued functions. Instead, we would like to track the set of possible values for such functions through action. Each K_x formula is also restricted to a set of literals that must be explicitly enumerated as a finite disjunction. Interval-valued functions provide a more compact representation that permits continuous intervals over number systems such as the reals ($\mathbb{R}$), which cannot be modelled in K_x.

Interval-valued functions can also be included in K_x, however, and are treated in the same way as any other piece of K_x information. In particular, this means they are subject to the conservative update rules inherent in that database. We refer the reader to (Petrick and Bacchus 2004) for more information about K_x and its update rules.[2]

Primitive queries and intervals The underlying primitive query language used by PKS is unchanged with the addition of interval-valued functions. In particular, the existing query language already permits primitive queries that include numeric relational operators such as those we permit in the extended K_w database (e.g., $>$). However, we have also extended the inference procedure that evaluates primitive queries to reason with interval-based information. For instance, a query of the form $K(f > 3)$ only evaluates as true given an interval $f = \langle u, v\rangle \in K_f$ provided $u > 3$. Similarly, a query $K(g \neq 5)$ is true if both $5 < u$ and $5 > v$.

Actions and intervals Actions in our extended version of PKS are defined in a similar way to ordinary PKS actions, with preconditions and effects. Preconditions are still simply sets of primitive queries, as defined above. Effects permit updates to be made to interval-valued information through a set of simple arithmetic operations. (In this paper we only consider the arithmetic addition and subtraction operators.) In particular, we allow updates to have the form

$$\mathit{add}(K_f, f := f \pm d),$$

where f is an interval-based function and d is either a numeric constant or constant interval (i.e., no variables). In the case of a constant d, an existing interval $\langle u, v\rangle$ is updated to the resulting interval $\langle u \pm d, v \pm d\rangle$. If d itself is an interval, the process is somewhat more complicated and a new range must be calculated for the resulting interval. For instance, adding the interval $\langle 3, 5\rangle$ to $\langle 1, 2\rangle$ results in an interval $\langle 4, 7\rangle$. We currently focus on arithmetic operations that can be calculated in a straightforward manner and result in well-formed intervals.

One additional update is performed when interval-valued updates occur: K_v formulae that are specified using interval schema are also updated appropriately. That is, the interval corresponding to a K_v formula is updated in a similar manner to an ordinary interval-based function. For instance, if $f : \langle x - c, x + c\rangle \in K_v$, and an effect of the form $\mathit{add}(K_f, f := f + d)$ updates f, where d is a constant, then K_v is updated so that $f : \langle x - c + d, x + c + d\rangle \in K_v$.

Finally, we also allow actions to include ordinary database assertions, following the standard PKS rules for *add* and *del*. Thus, we can specify "noisy" knowledge through an update such as $\mathit{add}(K_f, f = \langle 3, 5\rangle)$ that adds $f = \langle 3, 5\rangle$ to K_f.

PKS planning with intervals Given the above changes to the PKS database representation, primitive query mechanism, and database update procedure, the underlying planning algorithm operates as in the unextended version of PKS. In particular, the plan generation process is treated as a search through the set of database states, starting from an initial state denoted by the initial set of databases. Plans are built in a forward-chaining manner by choosing an action to

[2] One of the open technical questions in this work is whether or not intervals of the form $f = \langle 3, 5\rangle$ actually belong in K_f, or whether a better intuitive definition would place such knowledge in an extended K_x database. We leave open the possibility of changing our current representation in the future.

Action	Effects
moveForward	$add(K_f, robotLoc := robotLoc - 1)$
moveBackward	$add(K_f, robotLoc := robotLoc + 1)$
atTarget	$add(K_w, robotLoc = targetLoc)$

Table 1: Action specifications for Example 1.

add to a plan, or by introducing conditional plan branches. Planning continues until the goal conditions are achieved along every branch of a plan, or no plan can be found. We refer the reader to (Petrick and Bacchus 2002) for more details on the actual plan generation process used by PKS.

Examples

To illustrate the above extensions, we present three simple examples of interval-based reasoning in PKS.

Example 1 Consider a robot whose location, *robotLoc*, is measured by its distance to a wall. The robot has two physical actions available to it: *moveForward* moves the robot one unit towards the wall, and *moveBackward* moves the robot one unit away from the wall. The robot also has a sensing action, *atTarget*, which senses whether the robot is at a target location specified by the function *targetLoc*. The definition of these actions is shown in Table 1. The robot's initial location is specified by the interval-valued function mapping $robotLoc = \langle 3, 5\rangle \in K_f$. The goal is to move the robot to the target location, denoted by the query $K(robotLoc = targetLoc)$. In this example, $targetLoc = 2 \in K_f$.

One solution generated by PKS is the conditional plan:

```
1 | moveForward ;
2 | atTarget ;
3 | branch(robotLoc = targetLoc)
4 | K+ : nop.
5 | K− : moveForward ;
6 |      atTarget ;
7 |      branch(robotLoc = targetLoc)
8 |      K+ : nop.
9 |      K− : moveForward.
```

In step 1, the *moveForward* action uniformly decreases the value of *robotLoc* in K_f by one unit so that $robotLoc = \langle 2, 4\rangle$. In step 2, *atTarget* senses whether $robotLoc = targetLoc$, which has the effect of adding $robotLoc = 2$ to K_w (i.e., the planner knows whether *robotLoc* is 2 or not). In step 3, a branch point is added to the plan based on this K_w formula, allowing the plan to consider the two possible outcomes of the K_w formula (which also has the effect of removing the formula from K_w). Along one branch (step 4), $robotLoc = 2$ is assumed to be true (i.e., $robotLoc = 2$ is added to K_f) and the goal is achieved. Along the other branch (step 5), $robotLoc \neq 2$ is assumed to be true (i.e., $robotLoc \neq 2$ is added to K_f). As a result, the interval mapping for *robotLoc* in K_f can be refined to remove 2 as a possible mapping, so that $robotLoc = \langle 3, 4\rangle$. The *moveForward* action then updates *robotLoc* further so that $robotLoc = \langle 2, 3\rangle$. The sensing action in step 6 again adds $robotLoc = 2$ to K_w. In step 7, another branch point is added to the plan. Along one branch (step 8), $robotLoc = 2$ is assumed to true, satisfying the goal. Along the other branch (step 9), $robotLoc \neq 2$ is assumed to be true. In this case, refining *robotLoc* results in the (definite) knowledge that $robotLoc = \langle 3, 3\rangle = 3$. A final *moveForward* action results in $robotLoc = 2$, satisfying the goal.

Action	Effects
noisyForward	$add(K_f, robotLoc := robotLoc - \langle 1, 2\rangle)$
withinTarget	$add(K_w, robotLoc \leq targetLoc)$

Table 2: Additional actions for Example 2.

Example 2 We next consider a robot with the *moveBackward* and *atTarget* actions from Example 1, but with *moveForward* replaced by a "noisy" movement action, *noisyForward*, which moves the robot forward either 1 or 2 units. Additionally, the robot also has a second sensing action, *withinTarget*, that determines whether or not the robot is within the target distance (where $targetLoc = 2 \in K_f$). The specification of these new actions is given in Table 2. In this example, the robot's initial location is specified by the interval-valued mapping $robotLoc = \langle 3, 4\rangle \in K_f$. The goal is to move the robot to the target location, i.e., $K(robotLoc = targetLoc)$.

One solution generated by PKS is the conditional plan:

```
1  | noisyForward ;
2  | withinTarget ;
3  | branch(robotLoc ≤ targetLoc)
4  | K+ : atTarget ;
5  |      branch(robotLoc = targetLoc)
6  |      K+ : nop.
7  |      K− : moveBackward.
8  | K− : noisyForward ;
9  |      atTarget ;
10 |      branch(robotLoc = targetLoc)
11 |      K+ : nop.
12 |      K− : moveBackward.
```

Since forward movements may change the robot's position by either 1 unit or 2 units, the *noisyForward* action in step 1 results in an even less certain position for the robot, namely that $robotLoc = \langle 1, 3\rangle \in K_f$. However, the sensing action in step 2, together with the branch point in step 3, lets us split this interval into two sub-intervals. In step 4, we assume that $robotLoc \leq 2$ and consider the case that $robotLoc = \langle 1, 2\rangle$. The *atTarget* action, together with the branch in step 5, lets us divide this interval even further: in step 6, $robotLoc = 2$ and the goal is satisfied, while in step 7, $robotLoc = 1$ and a *moveBackward* action achieves the goal. In step 8 we consider the other sub-interval of the first branch, namely the interval resulting from $robotLoc > 2$, i.e., $robotLoc = 3 \in K_f$. In this case we have definite knowledge of the robot's location, however, the subsequent *noisyForward* action results in $robotLoc = \langle 1, 2\rangle$. The remainder of the plan in steps 9–12 is the same as in steps 4–7: the robot conditionally moves backwards in the case that *robotLoc* is determined to be 1, while the plan trivially achieves the goal if $robotLoc = 2$.

Example 3 In the final example, we consider a robot with the *moveBackward* action from Table 1, and the *noisyLocation* action from Table 3. In this case, *noisyLocation* is a noisy sensing action that either senses the actual value of the

Action	Effects
noisyLocation	*add*$(K_v, \textit{robotLoc} : \langle x, x+1 \rangle)$

Table 3: Additional action for Example 3.

robot's location, or 1 unit more than the actual location. This is denoted by the notation $\langle x, x+1 \rangle$ in the action description, where x acts as a placeholder for the actual location, and the interval specifies the range of possible values. Initially, the location of the robot is unknown, i.e., *robotLoc* is not listed in the planner's databases. The goal is to ensure the robot has moved to or past the target location, i.e., $K(\textit{robotLoc} \geq \textit{targetLoc})$, where $\textit{targetLoc} = 2 \in K_f$.

Here, PKS can generate the simple 3-step plan:

1 | *noisyLocation* ;
2 | *moveBackward* ;
3 | *moveBackward*.

Step 1 of the plan adds the formula $\textit{robotLoc} = \langle x, x+1 \rangle$ to K_v, indicating that the planner has (noisy) knowledge of the robot's location. In step 2, the result of *moveBackward* updates the planner's parametrized K_v knowledge. In particular, $\textit{robotLoc} = \langle x+1, x+2 \rangle$, which has the effect of tracking the movement action in relation to the planner's (ungrounded) location information. In step 3, the second *moveBackward* action results in $\textit{robotLoc} = \langle x+2, x+3 \rangle$. In this case, the planner can reason that $\textit{robotLoc} \geq 2$ holds since *robotLoc* is a function over $\mathbb{N}$: since $x \geq 0$, it must be the case that $x + 2 \geq 2$.

Although the above examples are admittedly simple, they nevertheless demonstrate some interesting plan-time reasoning. In Example 1, we illustrate a case where the planner has uncertain knowledge about the location of a robot. Using interval-valued functions, we track the robot's location as physical actions change this information and sensing actions, together with conditional plan branches, produce more certain knowledge. We note that in the original version of PKS, we could represent the disjunctive nature of *robotLoc* (for $\mathbb{N}$ at least) using the K_x database, e.g., $(\textit{robotLoc} = 3 \mid \textit{robotLoc} = 4 \mid \textit{robotLoc} = 5) \in K_x$. However, an action like *moveForward* would immediately invalidate this information, causing it to be removed, since it changes a function mentioned in the K_x formula.

In Example 2, we consider a simple case of an action with a noisy physical effect, represented by the action *noisyForward*, that changes *robotLoc* with an interval-based effect. Again, we demonstrate how the use of sensing actions together with conditional branching allows us to perform a type of case-based reasoning, to subdivide intervals into more manageable components. Of course, this example also demonstrates that if we do not have the *right* sensing actions (e.g., *withinTarget*), then the ability to track certain intervals alone may not always be sufficient for useful reasoning.

Finally, in Example 3 we illustrate an interesting type of reasoning we are currently experimenting with: the ability to track sensed information through physical actions using a type of placeholder variable. In particular, *noisyLocation* is a noisy sensing action whose resulting (indefinite) knowledge is tracked through *moveBackward* actions. We note that in this example the K_v interval is not strictly necessary for finding a plan since only the left endpoint of the interval is used. (I.e., an action that adds $\textit{robotLoc} : \langle x, x \rangle$ to K_v using a point interval would be sufficient.) However, the interval-based sensing action demonstrates one of the new features of our extended PKS representation.

Example 3 also demonstrates one of the drawbacks inherent in plan-time sensing with unknown quantities: for such values to be useful in a plan we often need to "ground" them in some way. In this case, we use knowledge of the underlying number system and the interval offset to make an assertion about a lower bound. However, PKS also has the ability to work with functions in an "unground" form, allowing them to be composed with other functions, or using them to produce a form of parametrized plan. One of the goals of this work is to extend PKS's representation of interval-valued functions so they can also be used in this way. One particular application where we believe this will be useful is in the automatic generation of plans with *loops* (Levesque 2005). (For instance, in the case of Example 3 we could imagine generating a parametrized plan that loops until a certain exit condition is achieved.) However, this extension is part of ongoing work.

Discussion and Future Work

Interval-valued functions provide an interesting middle ground between those representations that do not represent uncertainty about numerical values and those that reason with full possible-world models, or models based on probabilistic distributions. For a planner like PKS that works with a restricted representation to model particular types of knowledge, an interval-valued representation makes a good fit and offers another useful tool for knowledge-level planning. Moreover, such extensions have not been fully explored in the context of planning with incomplete information, sensing actions, and contingent plans, and this work offers the prospect of results that can be applied beyond PKS.

There are still some non-trivial technical problems to overcome. First, we are considering interval-based operations other than addition and subtraction. While we do not want to integrate a complete equation solver with our planner, we are focusing on those operations that are useful for planning and that can easily be "tracked" in simple, predictable ways. To help guide our work, we are investigating planning problems based on real-world robot domains requiring numeric reasoning. We are also exploring how existing approaches in the literature use intervals in other contexts (e.g., temporal reasoning).

Second, although intervals provide a compact means of representing a range of possible values, there are problems when those values are sparsely distributed. For instance, if f can map to the values 5, 7, and 10, then the interval $f = \langle 5, 10 \rangle$ suffices for representing the set of possible mappings, but also admits 6, 8, and 9 as possible values. While such intervals may not be problematic, depending on the task, they are potentially less accurate and may incur more reasoning than needed. To overcome these potential drawbacks, we are also investigating functions that map to *interval sets* consisting of a finite number of intervals. In

such cases, a function f could map to an interval set of the form $f = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n\}$, where each $\mathcal{I}_i$ is an interval. In PKS, this would combine the interval-based representation described here in a form of "extended-K_x" knowledge. To avoid situations of excessive fragmentation, with large numbers of intervals, we will initially bound the number of intervals allowed in a set, and in some cases use a single wide interval in the place of multiple intervals, if necessary.

Finally, we have not focused on the efficiency of PKS's plan generation process in this paper but have instead considered particular representation and reasoning problems. (We note that all the examples presented in this paper can be generated in less than a second using PKS on a single CPU running at 1.86 GHz with 2Gb of RAM.) In other work, we are also addressing the problem of scaling up PKS's performance by adapting heuristic search techniques to the state spaces produced by PKS. While interval-based representations may complicate this process somewhat, we believe that the compilation techniques of (Petrick 2006) could be adapted to this problem, allowing interval-based functions to be treated in a similar fashion to ordinary functions.

This paper presents a snapshot of work in progress. It also forms part of a larger research agenda aimed at transforming standard contingent planning domains (e.g., domains that can be represented in planners like Contingent-FF (Hoffmann and Brafman 2005)) into knowledge-level forms. It also builds on theoretical work in the situation calculus (Funge 1998; Demolombe and Pozos Parra 2000; Petrick 2006) that we are currently extending, with a focus on the construction of practical planning systems.

Acknowledgements

This work was partially funded by the Natural Sciences and Engineering Research Council (NSERC) scholarship program while the author was at the University of Toronto, and by the European Commission through the JAMES (Grant No. 270435) and Xperience (Grant No. 270273) projects.

References

Demolombe, R., and Pozos Parra, M. P. 2000. A simple and tractable extension of situation calculus to epistemic logic. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS-2000)*, 515–524.

Edelkamp, S. 2002. Taming numbers and durations in the model checking integrated planning system. *Journal of Artificial Intelligence Research* 20:195–238.

Etzioni, O.; Hanks, S.; Weld, D.; Draper, D.; Lesh, N.; and Williamson, M. 1992. An approach to planning with incomplete information. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, 115–125.

Etzioni, O.; Golden, K.; and Weld, D. 1994. Tractable closed world reasoning with updates. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, 178–189.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Frank, J., and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Journal of Constraints, Special Issue on Constraints and Planning* 8:339–364.

Funge, J. 1998. Interval-valued epistemic fluents. In *AAAI Fall Symposium on Cognitive Robotics*, 23–25.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-2005)*, 71–80.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.

Laborie, P. 2003. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143:151–188.

Levesque, H. J. 2005. Planning with loops. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, 509–515.

Liu, Y., and Levesque, H. J. 2005. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, 522–527.

Pednault, E. P. D. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, 324–332.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, 212–221.

Petrick, R. P. A., and Bacchus, F. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, 2–11.

Petrick, R. P. A. 2006. *A Knowledge-level approach for effective acting, sensing, and planning*. Ph.D. Thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Poggioni, V.; Milani, A.; and Baioletti, M. 2003. Managing interval resources in automated planning. *Journal of Information Theories and Applications* 10:211–218.

Soutchanski, M. 2001. A correspondence between two different solutions to the projection task with sensing. In *Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2001). http://www.cs.nyu.edu/faculty/davise/commonsense01/.*

Vassos, S., and Levesque, H. 2007. Progression of situation calculus action theories with incomplete information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2029–2034.