Action Selection for Interaction Management: Opportunities and Lessons for Automated Planning*

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

Mary Ellen Foster
School of Computing Science
University of Glasgow
Glasgow G12 8RZ, Scotland, UK
MaryEllen.Foster@glasgow.ac.uk

Abstract

The central problem in automated planning—action selection—is also a primary topic in the dialogue systems research community, however, the nature of research in that community is significantly different from that of planning, with a focus on end-to-end systems and user evaluations. In particular, numerous toolkits are available for developing speech-based dialogue systems that include not only a method for representing states and actions, but also a mechanism for reasoning and selecting the actions, often combined with a technical framework designed to simplify the task of creating end-to-end systems. We contrast this situation with that of automated planning, and argue that the dialogue systems community could benefit from some of the directions adopted by the planning community, and that there also exist opportunities and lessons for automated planning.

Introduction

At a basic level, the core automated planning problem is one of context-dependent action selection: given a set of initial conditions, a domain description (including a set of actions), and a set of goals, generate a sequence of actions whose application will bring about the goal conditions. However, the problem of action selection is not unique to the planning community. One important field where this issue is of primary concern is the *dialogue systems* community, a subfield of natural language dialogue that is focused on implementing tools and applications for interacting with human users.

A fundamental component of any dialogue system is the *interaction manager* (Bui 2006), whose primary task is to carry out a form of action selection: based on the current state of the interaction and of the world, the interaction manager makes a high-level decision as to which spoken, non-verbal, and task-based actions should be taken next by the system as a whole. In contrast to more formal, descriptive accounts of dialogue (Asher and Lascarides 2003), which aim to model the full generality of language use, work on interaction management has concentrated primarily on developing end-to-end systems and on evaluating them with human users (Jokinen and McTear 2009).

An important component of dialogue research has been the development of toolkits to support the construction of end-to-end systems. Such toolkits generally incorporate three main features. First, they provide a representational formalism for specifying states and actions. Second, the state/action representations are usually tightly linked to the reasoning strategy used to carry out action selection. Finally, most toolkits include infrastructure building tools to support modular system development. While these features can clearly simplify the task of implementing end-to-end systems, the fact that the features are so tightly connected complicates the task of comparing representational formalisms or reasoning strategies: in general, to carry out such a comparison, there is no alternative but to re-implement the entire system in other frameworks (Peltason and Wrede 2011).

In this paper, we argue that the dialogue community could benefit from the wider use of techniques that break this tight connection between action selection, representation, and technical architectures. As motivation for this view, we use the automated planning community as an example of a related field whose research directions have resulted in significantly different approaches to similar problems. While the central planning problem is also one of action selection, the planning community has focused on defining domains in common representation languages like PDDL (McDermott et al. 1998), and on comparing different action-selection strategies within this common context, especially through events like the International Planning Competitions (Coles et al. 2012); the study of the representation languages themselves has also led to a better understanding of the trade-offs between different representations (Rintanen 2004).

However, the exploration of dialogue systems also presents some opportunities and lessons for the planning community. First, the presence of action selection at the core of interaction management offers the obvious possibility of applying automated planning techniques in that community. Second, the nature of the problems addressed by dialogue systems also highlights the importance of applications—an area that has gained wider traction in the planning community but one that is still somewhat outside the mainstream of most planning research. Finally, the evaluation process for dialogue systems, and in particular the role of human users, may also present new directions for planning.

In the remainder of this paper, we discuss some of the

*This paper is the counterpart of (Foster and Petrick 2016), which was recently presented at the International Workshop on Spoken Dialogue Systems (IWSDS 2016).

```
rule( integrateUsrAnswer, [
    $/shared/lu/speaker = usr,
    assoc( $/shared/lu/moves, answer(R), false ),
    fst( $/shared/qud, Q ),
    $domain : relevant_answer( Q, R ),
    $domain : reduce(Q, R, P)
    ], [
    set_assoc( /shared/lu/moves, answer(R),true),
    shared/qud := $$pop( $/shared/qud ),
    add( /shared/com, P ) ] ).
```

Figure 1: Sample TrindiKit update rule (from http://www.ling.gu.se/~sl/Undervisning/Dialogsystem2_vt03/diasys2-trindikit.ppt).

available interaction management toolkits, summarising the representation, reasoning, and technical facilities provided by each. We then outline certain research directions from the automated planning community that could be of interest to dialogue systems work. Finally, we discuss the potential benefits to both the dialogue systems and automated planning communities that could result from a closer collaboration between these two research areas.

A Survey of Interaction Management Toolkits

In this section, we describe a representative set of dialogue systems toolkits, including several well-established ones and more recent developments. We primarily focus on the representations, including some examples, but also highlight the reasoning mechanisms and relevant details of the accompanying technical architecture.

TrindiKit/DIPPER: One of the most widely used approaches to dialogue management is the Information State Update approach, which is exemplified by TrindiKit (Larsson and Traum 2000) and its lighter-weight Java reimplementation DIPPER (Bos et al. 2003). The core of this approach is the use of an *information state* which represents the state of the dialogue and which is updated by applying *update rules* (see, e.g., Figure 1) following a given *update strategy*. The details of an information state are determined by the needs of a particular application. For example, the information state might include external aspects such as variables and their assignments (as in a slot-filling dialogue), or it might include internal agent states such as goals and beliefs (for a more plan-based dialogue strategy). TrindiKit and DIPPER both make use of the Open Agent Architecture (OAA) (Martin, Cheyer, and Moran 1999), which provides a middleware for integrating software agents into a distributed system. A similar Information State Update approach has also been taken in more recent dialogue systems, but using other infrastructures (Johnston et al. 2002; Janarthanam et al. 2015).

Ravenclaw: Another widely-used toolkit is Ravenclaw (Bohus and Rudnicky 2009), which is based around a *dialogue task specification* representing the domain-specific aspects of the control logic. This representation forms a hierarchical plan for the interaction and is executed by a domain-independent engine at run time. The specification consists

```
DEFINE_AGENCY( CPerformTask,
  DEFINE_CONCEPTS(
    INT_USER_CONCEPT(query_type, "")
    STRING_USER_CONCEPT(origin_place, "")
    STRING_USER_CONCEPT(destination_place, "")
    CUSTOM_SYSTEM_CONCEPT(result, CResultConcept)
    CUSTOM_SYSTEM_CONCEPT(new_result, CResultConcept)
  )
  DEFINE_SUBAGENTS(
    SUBAGENT(GetQuerySpecs, CGetQuerySpecs, "")
    SUBAGENT(ProcessQuery, CProcessQuery, "")
    SUBAGENT(GiveResults, CGiveResults, "")
  )
)
```

Figure 2: Sample Ravenclaw task specification (from http://wiki.speech.cs.cmu.edu/olympus/index.php/Tutorial_1).

```
<t:task id="CoachOpening">
  <t:precondition> otherLastChosen('Opening', 'correctOpening')
    </t:precondition>
  <t:subtasks id="coachOpeningSubtasks">
    <t:step name="coach" task="CoachOpeningDialog"/>
    <t:applicable> getUserEmotion() == 'happy' </t:applicable>
  </t:subtasks>
  <t:subtasks id="coachAltOpeningSubtasks">
    <t:step name="coach" task="CoachAltOpeningDialog"/>
    <t:applicable> getUserEmotion() == 'sad' </t:applicable>
  </t:subtasks>
</t:task>
```

Figure 3: Sample DISCO recipe (from https://github.com/charlesrich/Disco/blob/master/examples/tardis/models/Coach.d4g.xml).

of a tree of *dialogue agents*, each of which handles a subtask of the dialogue (e.g., asking for the origin and destination for a journey, as in Figure 2). The dialogue engine traverses the tree in a depth-first order, putting agents from the tree onto an execution stack and removing them when they are completed. The agents are defined through C++ macros that communicate by exchanging user-defined data structures through a message-passing system.

COLLAGEN/DISCO: COLLAGEN (Rich and Sidner 1998) is a toolkit based on the *collaborative interface paradigm*, which assumes that a software agent is collaborating with a user to operate an application programme, with both agents communicating with each other as well as interacting with the application. COLLAGEN has been used to implement a range of interface agents, including ones for travel booking and for controlling a programmable thermostat. More recently, COLLAGEN has been extended into an open-source tool called DISCO (Rich and Sidner 2012), which combines hierarchical task networks (HTNs) with traditional dialogue trees to permit semi-automated dialogue authoring and dialogue structure reuse. The target scenario is specified as a collection of *recipes*—that is, rules for decomposing a goal into subgoals and for accomplishing those subgoals, as in Figure 3. In contrast to Ravenclaw, where the

dialogue flow must be specified, COLLAGEN/DISCO only needs a specification of the tasks; the dialogue is then generated automatically via a generic rule framework.

OpenDial: OpenDial (Lison 2015) is a domain-independent toolkit for developing spoken dialogue systems. Its primary goal is to support robust dialogue management, using a hybrid framework that combines logical and statistical approaches through probabilistic rules to represent the internal models of the framework. OpenDial also includes a Java-based blackboard architecture where all modules are connected to a central information hub which represents the dialogue state, along with a plugin framework allowing new modules to be integrated.

IrisTK: IrisTK (Skantze and Al Moubayed 2012) is a toolkit for the rapid development of real-time systems for face-to-face multi-party interaction which accompanies the Furhat robot head (Al Moubayed et al. 2012). IrisTK provides an XML-based scripting language for defining *statecharts* (Harel 1987) that map input events to output events depending on the system state, along with an event-based distributed architecture that allows a system to be built by integrating modules such as speech recognition/synthesis. It also incorporates pre-built modules for such common tasks.

Summary

As highlighted above, and as summarised in Table 1, each of the described toolkits provides a different representation of the information needed for action selection, including declarative update rules, statecharts, or the more procedural representations used by toolkits such as Ravenclaw and COLLAGEN. Each toolkit also incorporates its own reasoning mechanism to make use of the defined representation—in fact, often the representation and reasoning components are so tightly related that they cannot be fully disentangled. Finally, the majority of the toolkits described (except for COLLAGEN/DISCO) either provide or make use of a specific technical middleware framework. As a result, the task of choosing a toolkit generally also means adopting both its reasoning strategy and its associated technical infrastructure.

This diversity of approaches has had the result that while it is common to compare interaction management strategies within a single framework—for example, by comparing action-selection policies that are learnt from data against hand-coded policies (Keizer et al. 2013)—it is relatively uncommon to compare the representational ability and reasoning performance across different frameworks. Peltason and Wrede (2011) carried out this sort of cross-toolkit comparison in which the same interactive system was implemented using Ravenclaw, DIPPER, Collagen/DISCO, and PaMini (Peltason and Wrede 2010); more recently, Olaso et al. (2016) did a similar study comparing DISCO and Ravenclaw. In both studies, the comparison required the entire dialogue system to be implemented separately in each formalism, with no possibility of transferring any representations or reasoning components across the implementations.

Lessons from Planning

The general problem of selecting high-level actions for an intelligent agent is not unique to dialogue systems, but is a problem addressed in a variety of research communities including automated planning. In planning, the emphasis is on applying problem-solving techniques to find an ordered sequence of actions (a *plan*) that, when chained together, will transform an initial state into a new state where a set of specified goal objectives are achieved.[1] A *planning domain* definition provides a description of the symbols and actions used by the planner, with the goal of much planning research to build *domain-independent* planning systems that are able to solve a range of planning problems in a variety of domains, rather than just a single problem in a single domain.

One important feature of many planning approaches is that the tools developed usually support one of a number of common representation languages, such as PDDL (McDermott et al. 1998), PPDDL (Younes and Littman 2004), or RDDL (Sanner 2010), among others. Many of these languages have been developed or extended as part of the International Planning Competitions (IPC) (Coles et al. 2012),[2] and within the context of the International Conference on Automated Planning and Scheduling (ICAPS). Even when some planners implement their own representation languages which may differ (usually syntactically) from the standard planning languages, work is often performed to establish the relationship between such languages and the more common representations.

These activities have led to some important benefits for planning. First, by adopting common representations, the task of modelling a planning problem can be separated from the task of implementing an efficient engine for solving those problems. This allows different planning engines to be developed and directly compared, either quantitatively or qualitatively, on a common set of inputs (i.e., planning problems). Second, planning domains and planning engines can be shared, leading to the development of common benchmarks for future planning systems, as well as improving the baseline systems that can solve problems in these domains. In particular, the IPC has contributed greatly to these activities by creating and requesting new domains, which has in turn helped spur the development of more powerful planning tools. These activities have also resulted in a repository of planning domains which can be studied, analysed, and reused as necessary. Finally, the representation languages themselves—and the planning problems they support—can be studied and compared, leading to a better understanding of the complexity of particular classes of domains and problems (Rintanen 2004), and the tradeoffs of using one language over another. This work has close connections to related communities such as knowledge representation and reasoning (KR&R) and formal logics, and has resulted in some interesting research directions, such as a range of com-

[1] This differs somewhat from interaction management where the goal is (usually) to find the next system action, rather than a complete action sequence. Note, however, that a system that is able to achieve the latter can also be used in the former context.

[2] http://www.icaps-conference.org/index.php/Main/Competitions

Toolkit	Representation	Reasoning	Technical
Trindikit/DIPPER	Information state	Update/selection rules	Open Agent Architecture (C++)
COLLAGEN/DISCO	Recipes	Generic rule framework	Java API
Ravenclaw	Task tree, agenda	Tree traversal	C++ macros, message passing
OpenDial	Probabilistic rules	Event-based state update	Java-based blackboard architecture
IrisTK	XML state charts	Event-based state update	Java event-based distributed architecture

Table 1: Summary of dialogue systems toolkits considered.

pilation approaches which seek to transform more complex planning problems into simpler forms that can potentially be solved more efficiently using existing tools (Palacios and Geffner 2009; Albore, Palacios, and Geffner 2009).

We believe that similar approaches could be applied within the dialogue systems research community, leading to similar positive results. Indeed, concrete example of this approach do exist (Petrick and Foster 2013) as valuable case studies, however, such work remains outside the mainstream of dialogue systems research.

Opportunities and Lessons for Planning

One of the obvious opportunities for planning—where domains have long been defined in common representation languages, and action-selection strategies compared within this common context—is that dialogue systems present an opportunity for showcasing planning tools and demonstrating how different approaches can be benchmarked and compared more easily.[3] Although early work was done in this area (Perrault and Allen 1980; Appelt 1985; Hovy 1988; Cohen and Levesque 1990; Young and Moore 1994), the approach has for the most part been largely overlooked more recently (with the exception of approaches like (Koller and Stone 2007; Benotti 2008; Brenner and Kruijff-Korbayová 2008)). Problems in dialogue systems can also serve as the basis for new challenge domains for planning, possibly extending the standard planning representations with the features needed to model new types of problems.

However, beyond the opportunity for novel test domains, there are also some important general lessons that the planning community can learn from the dialogue systems area. For instance, dialogue systems are inherently application driven and as such, any adoption of planning techniques must be situated in the context of larger, more complex systems of which planning is a single component. This often requires a degree of maturity in tool development that goes beyond offline lab-tested code, with a focus on robustness and the development of standard application programming interfaces (APIs). While there have been recent attempts to build such systems within the planning community (Cashmore et al. 2015), more work is still needed.

Moreover, the issue of user evaluation is at the heart of dialogue systems work, with a focus on (non-expert) users actually using the developed tools. While issues like planning time and plan quality (the standard planning metrics) are clearly important to this problem, other factors related to the issue of online execution (e.g., plan execution monitoring in unpredictable domains, and the ability of a planner to generate alternative plans) can also play a larger role in these settings. In particular, dialogue systems domains are often driven by the needs of the real-world application, rather than lab-based assumptions. As a result, a range of related problem areas also come to the forefront, including plan verification, plan explanation, plan visualisation, and interfaces for domain modification by non-experts.

[3] It is worth noting that common tasks such as the Dialogue State Tracking challenge (Henderson, Thomson, and Williams 2014) do exist in the dialogue community; however, to our knowledge, there has never been a successful effort to develop standard, high-level representations for use in interaction management.

Conclusion

Overall, we believe the time is right for closer links between the dialogue systems and automated planning communities, with an opportunity in the first instance for the dialogue systems community (and possibly, the wider natural language community) to benefit from recent advances arising from the planning community. In particular, the use of common, formally understood representation languages for states and actions, combined with standard tools that separate problem representation from reasoning mechanisms and technical infrastructure, can serve as the basis for closer ties between the two communities, with important opportunities and lessons for the planning community as well.

Towards this end, we plan to continue our own work on applying planning techniques to interaction management (Petrick and Foster 2013), which began with the JAMES project,[4] and to highlight the challenges and opportunities that arise from the intersection of these two communities.

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant no. 270435 (JAMES, `james-project.eu`) and grant no. 610917 (STAMINA, `stamina-robot.eu`).

[4] `http://james-project.eu/`

References

Al Moubayed, S.; Beskow, J.; Skantze, G.; and Granström, B. 2012. Furhat: A back-projected human-like robot head for multiparty human-machine interaction. In *Cognitive Behavioural Systems*, volume 7403 of *Lecture Notes in Computer Science*. 114–130.

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proceedings of IJCAI 2009*, 1623–1628.

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

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

Benotti, L. 2008. Accommodation through tacit sensing. In *Proceedings of LONDIAL 2008*, 75–82.

Bohus, D., and Rudnicky, A. I. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language* 23(3):332–361.

Bos, J.; Klein, E.; Lemon, O.; and Oka, T. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proc. of SIGdial 2003*, 115–124.

Brenner, M., and Kruijff-Korbayová, I. 2008. A continual multiagent planning approach to situated dialogue. In *Proceedings of LONDIAL 2008*, 67–74.

Bui, T. H. 2006. Multimodal dialogue management - state of the art. Technical Report 06–01, University of Twente (UT), Enschede, The Netherlands.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In *Proceedings of ICAPS 2015*.

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

Coles, A.; Coles, A.; García Olaya, A.; Jiménez, S.; Linares López, C.; Sanner, S.; and Yoon, S. 2012. A survey of the seventh international planning competition. *AI Magazine* 33(1):83–88.

Foster, M. E., and Petrick, R. P. A. 2016. Separating representation, reasoning, and implementation for interaction management. In *Proceedings of the Seventh International Workshop on Spoken Dialogue System (IWSDS 2016)*.

Harel, D. 1987. Statecharts: A visual formalism for complex systems. *Science of Computer Programming* 8:231–274.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014. The second dialog state tracking challenge. In *Proc. of SIGdial 2014*, 263–272.

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

Janarthanam, S.; Hastie, H.; Deshmukh, A.; Aylett, R.; and Foster, M. E. 2015. A reusable interaction management module: Use case for empathic robotic tutoring. In *Proceedings of goDIAL 2015*.

Johnston, M.; Bangalore, S.; Vasireddy, G.; Stent, A.; Ehlen, P.; Walker, M.; Whittaker, S.; and Maloor, P. 2002. MATCH: An architecture for multimodal dialogue systems. In *Proceedings of ACL 2002*, 376–383.

Jokinen, K., and McTear, M. 2009. Spoken dialogue systems. *Synthesis Lectures on Human Language Technologies* 2(1):1–151.

Keizer, S.; Foster, M. E.; Lemon, O.; Gaschler, A.; and Giuliani, M. 2013. Training and evaluation of an MDP model for social multi-user human-robot interaction. In *Proc. of SIGdial 2013*.

Koller, A., and Stone, M. 2007. Sentence generation as planning. In *Proceedings of ACL 2007*, 336–343.

Larsson, S., and Traum, D. R. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering* 6(3&4):323–340.

Lison, P. 2015. A hybrid approach to dialogue management based on probabilistic rules. *Computer Speech & Language*.

Martin, D. L.; Cheyer, A. J.; and Moran, D. B. 1999. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence* 13(1-2):91–128.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. 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.

Olaso, J. M.; Milhorat, P.; Himmelsbach, J.; Boudy, J.; Chollet, G.; Schlögl, S.; and Torres, M. I. T. 2016. A multi-lingual evaluation of the vAssist spoken dialog system: Comparing Disco and RavenClaw. In *Proceedings of the Seventh International Workshop on Spoken Dialogue System (IWSDS 2016)*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Peltason, J., and Wrede, B. 2010. PaMini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In *Proc. of SIGdial 2010*, 229–232.

Peltason, J., and Wrede, B. 2011. The curious robot as a case-study for comparing dialog systems. *AI Magazine* 32(4):85–99.

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

Petrick, R. P. A., and Foster, M. E. 2013. Planning for social interaction in a robot bartender domain. In *Proceedings of ICAPS 2013, Special Track on Novel Applications*.

Rich, C., and Sidner, C. L. 1998. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction* 8(3–4):315–350.

Rich, C., and Sidner, C. L. 2012. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Intelligent Virtual Agents*, volume 7502 of *Lecture Notes in Computer Science*. 327–340.

Rintanen, J. 2004. Complexity of planning with partial observability. In *Proceedings of ICAPS 2004*, 345–354.

Sanner, S. 2010. Relational dynamic influence diagram language (RDDL): Language description. `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`.

Skantze, G., and Al Moubayed, S. 2012. IrisTK: A statechart-based toolkit for multi-party face-to-face interaction. In *Proceedings of ICMI 2012*, 69–76.

Younes, H. L. S., and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-162, CMU.

Young, R. M., and Moore, J. D. 1994. DPOCL: a principled approach to discourse planning. In *Proc. of INLG 2004*, 13–20.