A New Approach to Planning in Networks

Jussi Rintanen
NICTA & the Australian National University
Canberra, Australia

Abstract. Control of networks like those for transportation, power distribution, communication to name a few, provides challenges to planning and scheduling. Many problems can be defined in terms of a basic state space model, but more general problems require an expressive language for talking about the topology and connectivity of the system, which are outside the scope of standard planning languages. In this work we introduce a general framework for defining planning languages for networked systems, with capability to express properties of connectivity and topology of such systems.

1 Introduction

Other areas of computer science that use the transition system model of actions include computer-aided verification and validation, where reachability analysis and model-checking problems are described in languages like SMV [6] and PROMELA [5]. These languages describe transition systems in terms concepts naturally occuring in the relevant application areas of the tools.

Much of the high-level technological infrastructure has the form of networks: transportation, telecommunications, power distribution and water distribution are all based on networks with clearly definable nodes and edges connecting them. Also, most of the standard planning benchmark problems involve networks. Even minor extensions to many of these problems are difficult to express compactly in standard planning languages.

For network-structured applications we propose the use of high-level planning languages with network features, as well as efficient algorithms for directly solving the problems expressed in these languages. A practical requirement for this approach to be feasible is that the overall complexity is not increased, in comparison to expressing the same problems in a standard classical planning language. A nominally "tractable" reduction of network-planning problems to standard planning languages is often possible, but this involves, in all but the simplest cases, a prohibitively high increase in the size of the problem instance and solution times.

To illustrate the differences between the approaches, consider a real-world planning problem with network structure that has been considered in earlier work on AI planning: the power-supply restoration problem for electricity distribution networks [7]. The first modeling of this problem in a classical planning language similar to PDDL leads to huge problem descriptions even for small networks [1]. Using the *axioms* of PDDL [4] to express network connectivity properties leads to a much more practical representation of the problem [2]. This formulation is compact but arguably not very natural as the axiom mechanism (inductive definitions) doesn't per se directly represent any natural features of this domain.

2 Problem Definition

We model systems in terms of a set of nodes and connections between them. The properties of each node are expressed in terms of state variables. Every node has the same set of state variables, but as the actions need not treat the nodes uniformly, this is not a restriction. In this paper we only consider a deterministic planning problem similarly to classical planning, and hence we have a unique initial state for the system.The state of the system consists of the connections and the values of the state variables.

Definition 1 (State) *For given sets A of state variables, V of nodes and E of (atomic) connections, a state is a pair (v, e) where*

- $v : V \times A \to \{0, 1\}$ *assigns a value to each state variable at each node, and*
- $e : E \to 2^{V \times V}$ *assigns (atomic) connections a binary relation.*

Definition 2 *A network is defined as (A, V, E, O, I, G) where*

- *A is the set of (Boolean)* state variables*,*
- *V is the set of* nodes*,*
- *E is the set of (atomic)* connections *between the nodes,*
- *O is the set of* actions *(to be defined later),*
- *I is the* the initial state*, and*
- *G is a modal formula representing the* goal *of the system (to be defined later.)*

2.1 Network Properties

We employ modal logic to express properties of systems. The modalities express connections between nodes.

- Atomic connections $c \in E$ are connections.
- If c_1 and c_2 are connections then so are $c_1; c_2$, $c_1 \cup c_2$ and $c_1 \cap c_2$.
- If c is a connection then so are $c*$ and c^{-1}.
- If ϕ is a formula then $\phi?$ is a connection.

Composite connection $c_1; c_2$ between nodes n and n' means that n' can be reached from n by first following c_1 to some intermediate node and from there c_2 to n'. Connection $c_1 \cup c_2$ expresses disjunctivity: there is either connection c_1 or c_2. Analogously, $c_1 \cap c_2$ expresses conjunctivity. The connection $c*$ represents the reflexive transitive closure of c. The connection c^{-1} is the inverse of c: a connection going from n' to n whenever c goes from n to n'. The connection $\phi?$ conditionally connects a node with itself if ϕ is true.

Definition 3 *The meaning $[\![c]\!]^S_s$ of connections c in a state $s = (v, e)$ of $S = (A, V, E, O, I, G)$ is defined as follows.*

- $[\![c]\!]^S_s = e(c)$ *if* $c \in E$
- $[\![c; c']\!]^S_s = \{(x,z) | (x,y) \in [\![c]\!]^S_s, (y,z) \in [\![c']\!]^S_s\}$
- $[\![c \cup c']\!]^S_s = [\![c]\!]^S_s \cup [\![c']\!]^S_s$
- $[\![c \cap c']\!]^S_s = [\![c]\!]^S_s \cap [\![c']\!]^S_s$
- $[\![c*]\!]^S_s = [\![c]\!]^S_s *$
- $[\![c^{-1}]\!]^S_s = \{(n,m) | (m,n) \in [\![c]\!]^S_s\}$
- $[\![\phi?]\!]^S_s = \{(t,t) \in V \times V | s \models_t \phi\}$

Here the operations $\cup$, $\cap$ *and* $*$ *on the right hand sides are the set-theoretic union and intersection and the reflexive transitive closure.*

The connections are used as a part of a modal language that includes the classical propositional logic. The atomic formulas include the propositional variables and the names of nodes.

- The constants $\bot$ and $\top$ (for *false* and *true*) are formulas.
- a for a state variable $a \in A$ is a formula.
- n for a node $n \in V$ is a formula.
- $\phi \vee \psi$ is a formula where ϕ and ψ are formulae.
- $\neg\phi$ is a formula where ϕ is a formula.
- $[c]\phi$ is a formula if ϕ is a formula and c is a connection.
- $n{:}\,\phi$ is a formula if ϕ is a formula and $n \in V$ is a node.

The modal operators $[c]$ represent universal quantification over all nodes that are reachable by a path described by c. The formula n is true in the node n and false elsewhere. Formulae $n{:}\,\phi$ refer to the truth of ϕ in node n. The meaning of $\rightarrow$ and $\wedge$ and $\leftrightarrow$ is defined in the usual way, as is $\langle c\rangle$ by $\langle c\rangle\phi = \neg[c]\neg\phi$.

Example 1 *The next formula is true in cities (nodes) from which one can fly to a tropical destination with a direct flight or two flights without changing planes in the U.S.*

$$\langle \mathit{flight} \cup (\mathit{flight}; \neg \mathit{US}?; \mathit{flight})\rangle \mathit{tropics}$$

The next formula is true if there are paths in a communications network from the current node to node n *that go through a designated* center *node and only visit nodes that are safe on the way.*

$$\langle (\mathit{link}; \mathit{safe}?)*; \mathit{center}?; (\mathit{safe}?; \mathit{link})*\rangle n$$ ■

At this point it is apparent that our logic is a variant of the propositional dynamic logic (PDL) [3] with a mechanism for referring to the names of nodes. A truth-definition for this modal logic can be given in the obvious way. We define $s \models \phi$ iff $s \models_n \phi$ for all $n \in V$.

2.2 Actions

Actions can change the values of the state variables associated with the nodes and the connections between the nodes.

An action consists of a *precondition* which determines the circumstances under which the action can be taken, as well as *the effects* that indicate when and how do the values of the state variables change and which connections between nodes are added or removed.

Definition 4 (Action) *An* action *is* $\langle p, e\rangle$ *where* p *is a formula and* e *is a set of pairs* $q \rhd r$ *where* q *is a formula and* r *is a set of literals. The literals that can be effects in an action are* $n : a$, $\neg n : a$ *and* (n, c, n') *and* $\neg(n, c, n')$*, where* $a \in A$, $n \in N$, $n' \in N$ *and* $c \in C$.

Definition 5 (Successor state) *Let* $S = (A, V, E, O, I, G)$ *be a system. Let* $\langle p, e\rangle$ *be an action and* $s = (v, g)$ *a state. The action is* executable *if* $s \models p$ *and the set* $F = \bigcup\{r | q \rhd r, s \models q\}$ *is consistent. The* successor state *of* s *is* $s' = (v', g')$ *where*

- $v'(n,a) = \begin{cases} 1 & \text{if } n{:}\,a \in F \\ 0 & \text{if } \neg n{:}\,a \in F \\ v(n,a) & \text{otherwise} \end{cases}$ *for all* $n \in V$ *and* $a \in A$
- $g'(c) = \begin{array}{l} g(c) \backslash \{(n,n') | \neg(n,c,n') \in F\} \\ \cup \{(n,n') | (n,c,n') \in F\} \end{array}$ *for all* $c \in E$

3 Examples

Many of the standard planning benchmark problems can be viewed as consisting of nodes and connections between them.

Example 2 *In Blocks World the blocks are the nodes and the on-relation are the connections. Moving* x *from* y *onto* z *is defined by*

$$\left\langle x{:}\,\langle \mathit{on}\rangle y \wedge x{:}\,[\mathit{on}^{-1}]\bot \wedge z{:}\,[\mathit{on}^{-1}]\bot, \top \rhd \{\neg(x, \mathit{on}, y), (x, \mathit{on}, z)\}\right\rangle.$$

Here $x{:}\,[\mathit{on}^{-1}]\bot$ *says that* false *is true in all nodes related to* x *by on, meaning that there is no such node, i.e., the block is clear.*

We can introduce an action that allows moving stacks of blocks.

$$\left\langle x{:}\,\langle \mathit{on}\rangle y \wedge z{:}\,\neg\langle \mathit{on}*\rangle x \wedge z{:}\,[\mathit{on}^{-1}]\bot, \top \rhd \{\neg(x, \mathit{on}, y), (x, \mathit{on}, z)\}\right\rangle$$ ■

Example 3 *With the network planning language it is easy to express movement from one node to any of the reachable nodes. Here* a *and* b *are names of locations and* p *is an object that moves.*

$$\langle a{:}\,(p \wedge \langle \mathit{road}*\rangle b), \top \rhd \{\neg a{:}\,p, b{:}\,p\}\rangle$$

Further extensions are possible. For example, we can require that property ϕ *is satisfied after each road segment along the path.*

$$\langle a{:}\,(p \wedge \langle (\mathit{road}; \phi?)*\rangle b), \top \rhd \{\neg a{:}\,p, b{:}\,p\}\rangle$$ ■

4 Conclusions

We have considered a language for planning in networks. The language is directly relevant to many application domains with network structure. The network model is even more general, allowing to express interesting properties of benchmarks that at the surface level are not about networks. This is a consequence of many problems having a relational/graph representation and the support of the language for expressing properties of graphs.

Acknowledgements

The research was funded by Australian Government's Department of Broadband, Communications and the Digital Economy and the Australian Research Council through NICTA and the SuperCom project.

REFERENCES

[1] P. Bertoli, A. Cimatti, J. K. Slaney, and S. Thiébaux, 'Solving power supply restoration problems with planning via symbolic model checking', in *ECAI'02*, pp. 576–580, (2002).

[2] B. Bonet and S. Thiébaux, 'GPT meets PSR', in *ICAPS'03*, pp. 102–112, (2003).

[3] Michael J. Fischer and Richard E. Ladner, 'Propositional dynamic logic of regular programs', *J. Computer and System Sciences*, **18**(2), 194–211, (1979).

[4] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, 'PDDL - the Planning Domain Definition Language, version 1.2', Technical report, Yale Center for Computational Vision and Control, Yale University, (1998).

[5] Gerald J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, 1991.

[6] Kenneth L. McMillan, *Symbolic Model Checking*, Kluwer, 1993.

[7] S. Thiébaux and M.-O. Cordier, 'Supply restoration in power distribution systems – a benchmark for planning under uncertainty', in *ECP'01*. Springer, (2001).