TU4

ICAPS 2006

The English Lake District, Cumbria, UK

Tutorial on Planning and Complexity

Malte Helmert

Albert-Ludwigs-Universität, Germany

icaps

ICAPS 2006

The English Lake District, Cumbria, UK

TU4

Tutorial on Planning and Complexity

Malte Helmert

Albert-Ludwigs-Universität, Germany

ICAPS 2006
Tutorial on Planning and Complexity

Table of contents

Preface

Planning is a hard problem. But just how hard, exactly? This tutorial addresses this question in a formal and rigorous manner using the tools of theoretical computer science. Why theoretical studies? Maybe the most important piece of information that complexity theory can provide the practitioner is advice on how not to solve a problem. For example, if it is known that a certain problem cannot be solved by Turing Machines with polynomial space requirements, then there is no point in trying to design an algorithm with that property. The tutorial introduces a number of such lower bounds on complexity for variants of the planning problem, not limited to the classical case of STRIPS-style planning, but discussing a wide spectrum of planning problems of progressing difficulty.

The tutorial is largely self-contained, and does not assume in-depth knowledge of complexity theory, although a certain familiarity with basic concepts like Turing Machines, reducibility and basic complexity classes like P and NP is definitely of help. It is targeted at an audience with a solid background on AI Planning, but little or no prior exposure to the theoretical work in the field.

The tutorial is structured into four parts.

Part 1, ”Foundations”, develops the necessary formal background in computational complexity that is required for the theoretical analyses that follow.

Part 2, ”Classical Planning”, applies these methods to the ubiquitous ”PDDL-style” planning problem, i.e., planning with full observability and complete determinism, both in the case where planning tasks are defined in terms of propositional variables and in the first-order case where planning tasks are defined in terms of predicates and schematic operators.

Part 3, ”Conditional Planning”, moves beyond the restrictions of the classical scenario by considering three generalizations thereof: planning with nondeterministic operators and full observability, planning with deterministic operators (but a nondeterministic initial state) and partial or no observability, and finally the most general case of planning with nondeterministic operators and partial observability, whose complexity has only been determined recently by Rintanen.

Part 4, ”Numeric Planning”, returns to the deterministic setting, but moves away from finite state spaces, by introducing numerical state variables. This variant of the planning problem is easily proved undecidable, so a number of restrictions are considered, providing a precise picture of what is and is not possible once exhaustive search is no longer an option.

Malte Helmert

Instructor

- *Malte Helmert, Albert-Ludwigs-Universitat Freiburg*

Planning and Complexity
Introduction

Malte Helmert

June 7th, 2006

Outline

1. Introduction
   - Motivation
   - Target Audience
   - Goals
   - Topics
   - Non-Topics
   - Literature

Why Complexity?

- understand the problem
- know what it is not possible
- find interesting subproblems
- distinguish essential features from syntactic sugar

Prerequisites

- basic knowledge of theoretical computer science
  - Turing Machines
  - basic complexity classes: P, NP, etc.
  - decidability
- basic knowledge of AI planning
  - STRIPS-style planning formalisms

Goals of the Tutorial

- present central complexity results
- show tradeoff expressivity vs. complexity
- demonstrate methodology for theoretical analyses

Overview of Topics

1. Foundations
   - Turing Machines
   - complexity classes
2. Classical Planning
   - propositional case
   - first-order case
3. Conditional Planning
   - nondeterministic operators, full observability
   - deterministic operators, no observability
   - nondeterministic operators, partial observability
4. Numeric Planning
   - undecidable cases
   - decidable cases

Some Relevant Non-Topics

- propositional planning with syntactic restrictions
- propositional planning with structural restrictions
- compilability between planning formalism
- domain-dependent planning complexity
- approximation results

Literature: Foundations

Christos H. Papadimitriou.
Computational Complexity.
Addison-Wesley, 1994.

Michael R. Garey and David S. Johnson.
Computers and Intractability —
A Guide to the Theory of NP-Completeness.
Freeman, 1979.

Giorgio Ausiello, Pierluigi Crescenzi, Giorgo Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi.
Complexity and Approximation.
Springer-Verlag, 1999.

Literature: Classical Planning

- Tom Bylander.
  The computational complexity of
  propositional STRIPS planning.
  Artificial Intelligence, 69(1–2):165–204, 1994.

- Kutluhan Erol, Dana S. Nau, and V. S. Subrahmanian.
  Complexity, decidability and undecidability results
  for domain-independent planning.
  Artificial Intelligence, 76(1–2):65–88, 1995.

Literature: Conditional Planning

- Patrik Haslum and Peter Jonsson.
  Some results on the complexity of planning
  with incomplete information.
  In *Proc. ECP-99*, pages 308–318, 1999.

- Jussi Rintanen.
  Complexity of planning with partial observability.
  In *Proc. ICAPS 2004*, pages 345–354, 2004.

- Jussi Rintanen.
  Planning: Algorithms and Complexity.
  Habilitation thesis, Albert-Ludwigs-Universität Freiburg, 2005.

Literature: Numeric Planning

Malte Helmert.
Decidability and undecidability results for planning with numerical state variables.
In *Proc. AIPS 2002*, pages 303–312, 2002.

Literature: Propositional Planning with Syntactic Restrictions

Tom Bylander.
The computational complexity of propositional STRIPS planning.
Artificial Intelligence, 69(1–2):165–204, 1994.

Literature: Propositional Planning with Structural Restrictions

- Christer Bäckström and Bernhard Nebel.
  Complexity results for SAS^+ planning.
  Computational Intelligence, 11(4):625–655, 1995.
- Peter Jonsson and Christer Bäckström.
  State-variable planning under structural restrictions: Algorithms and complexity.
  Artificial Intelligence, 100(1–2):125–176, 1998.
- Peter Jonsson and Christer Bäckström.
  Tractable plan existence does not imply tractable plan generation.
  Annals of Math. and AI, 22(3):281–296, 1998.

Literature: Propositional Planning with Structural Restrictions (continued)

- Carmel Domshlak and Yefim Dinitz.
  Multi-agent off-line coordination: Structure and complexity.
  In *Proc. ECP 2001*, pages 277–288, 2001.
- Carmel Domshlak and Ronen I. Brafman.
  Structure and complexity in planning with unary operators.
  In *Proc. AIPS 2002*, pages 34–43, 2002.

Literature: Compilability Between Planning Formalisms

- Bernhard Nebel.
  What is the expressive power of disjunctive preconditions?
  In *Proc. ECP-99*, pages 294–307, 1999.

Literature: Domain-Dependent Planning Complexity

- Naresh Gupta and Dana S. Nau.
  On the complexity of blocks-world planning.
  Artificial Intelligence, 56(2–3):223–254, 1992.
- Bart Selman.
  Near-optimal plans, tractability, and reactivity.
  In *Proc. KR-94*, pages 521–529, 1994.
- John Slaney and Sylvie Thiébaux.
  Blocks World revisited.
  Artificial Intelligence, 125:119–153, 2001.

Literature: Domain-Dependent Planning Complexity (continued)

- Malte Helmert.
  Complexity results for standard benchmark domains in planning.
  Artificial Intelligence, 143(2):219–262, 2003.
- Malte Helmert.
  New complexity results for classical planning benchmarks.
  In *Proc. ICAPS 2006*.

Literature: Approximation Results

- Bart Selman.
  Near-optimal plans, tractability, and reactivity.
  In *Proc. KR-94*, pages 521–529, 1994.
- Malte Helmert, Robert Mattmüller, and Gabi Röger.
  Approximation properties of planning benchmarks.
  In *Proc. ECAI 2006*.
  To appear.

Planning and Complexity
Part 1: Foundations

Malte Helmert

June 7th, 2006

Outline

Alternating Turing Machines

Definition: Alternating Turing Machine

Alternating Turing Machine (ATM) $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$:

1. input alphabet Σ and blank symbol $\Box \notin \Sigma$
   - alphabets always non-empty and finite
   - tape alphabet $\Sigma_\Box = \Sigma \,\dot\cup\, \{\Box\}$
2. finite set Q of internal states with initial state $q_0 \in Q$
3. state labeling $l : Q \to \{\mathsf{Y}, \mathsf{N}, \exists, \forall\}$
   - accepting, rejecting, existential, universal states Q_Y, Q_N, $Q_\exists$, $Q_\forall$
   - terminal states $Q_\star = Q_\mathsf{Y} \cup Q_\mathsf{N}$
   - nonterminal states $Q' = Q_\exists \cup Q_\forall$
4. transition relation $\delta \subseteq (Q' \times \Sigma_\Box) \times (Q \times \Sigma_\Box \times \{-1, +1\})$

(Non-) Deterministic Turing Machines

Definition: Non-deterministic Turing Machine

A non-deterministic Turing Machine (NTM) is an ATM where all nonterminal states are existential.

- no universal states

Definition: Deterministic Turing Machine

A deterministic Turing Machine (DTM) is an NTM where the transition relation is functional.

- for all $(q, a) \in Q' \times \Sigma_\Box$, there is exactly one triple (q', a', Δ) with $((q, a), (q', a', \Delta)) \in \delta$
- notation: $\delta(q, a) = (q', a', \Delta)$

Turing Machine Configurations

Let $M = \langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ be an ATM.

Definition: Configuration

A configuration of M is a triple $(w, q, x) \in \Sigma_\Box^* \times Q \times \Sigma_\Box^*$.

- w: tape contents before tape head
- q: current state
- x: tape contents after and including tape head

Turing Machine Transitions

Let $M = \langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ be an ATM.

Definition: Yields relation

A configuration c of M yields a configuration c' of M, in symbols $c \vdash c'$, as defined by the following rules, where $a, a', b \in \Sigma_\Box$, $w, x \in \Sigma_\Box^*$, $q, q' \in Q$ and $((q, a), (q', a', \Delta)) \in \delta$.

$$
\begin{aligned}
(w, q, ax) &\vdash (wa', q', x) && \text{if } \Delta = +1, |x| \geq 1 \\
(w, q, a) &\vdash (wa', q', \Box) && \text{if } \Delta = +1 \\
(wb, q, ax) &\vdash (w, q', ba'x) && \text{if } \Delta = -1 \\
(\epsilon, q, ax) &\vdash (\epsilon, q', \Box a'x) && \text{if } \Delta = -1
\end{aligned}
$$

Acceptance (Time)

Let $M = \langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ be an ATM.

Definition: Acceptance (time)

Let (w, q, x) be a configuration of M.

- M accepts (w, q, x) with $q \in Q_{\mathsf{Y}}$ in time n for all $n \in \mathbb{N}_0$.
- M accepts (w, q, x) with $q \in Q_{\exists}$ in time n iff M accepts some c' with $c \vdash c'$ in time $n - 1$.
- M accepts (w, q, x) with $q \in Q_{\forall}$ in time n iff M accepts all c' with $c \vdash c'$ in time $n - 1$.

Acceptance (Space)

Let $M = \langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ be an ATM.

Definition: Acceptance (space)

Let (w, q, x) be a configuration of M.

- M accepts (w, q, x) with $q \in Q_{\mathsf{Y}}$ in space n iff $|w| + |x| \leq n$.
- M accepts (w, q, x) with $q \in Q_{\exists}$ in space n iff M accepts some c' with $c \vdash c'$ in space n.
- M accepts (w, q, x) with $q \in Q_{\forall}$ in space n iff M accepts all c' with $c \vdash c'$ in space n.

Accepting Words and Languages

Let $M = \langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ be an ATM.

Definition: Accepting words

M accepts the word $w \in \Sigma^*$ in time (space) $n \in \mathbb{N}_0$
iff M accepts (ϵ, q_0, w) in time (space) n.

Definition: Accepting languages

Let $f : \mathbb{N}_0 \to \mathbb{N}_0$.
M accepts the language $L \subseteq \Sigma^*$ in time (space) f
iff M accepts each word $w \in L$ in time (space) $f(|w|)$,
and M does not accept any word $w \notin L$.

Outline

Time Complexity

Definition: DTIME, NTIME, ATIME

Let $f : \mathbb{N}_0 \to \mathbb{N}_0$.

Complexity class DTIME(f) contains all languages accepted in time f by some DTM.

Complexity class NTIME(f) contains all languages accepted in time f by some NTM.

Complexity class ATIME(f) contains all languages accepted in time f by some ATM.

Space Complexity

Definition: DSPACE, NSPACE, ASPACE

Let $f : \mathbb{N}_0 \to \mathbb{N}_0$.

Complexity class DSPACE(f) contains all languages accepted in space f by some DTM.

Complexity class NSPACE(f) contains all languages accepted in space f by some NTM.

Complexity class ASPACE(f) contains all languages accepted in space f by some ATM.

Polynomial Complexity Classes

Let $\mathcal{P}$ be the set of polynomials $p : \mathbb{N}_0 \to \mathbb{N}_0$.

Definition: P, NP, . . .

$$\begin{aligned}
\mathsf{P} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{DTIME}(p)\\
\mathsf{NP} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{NTIME}(p)\\
\mathsf{AP} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{ATIME}(p)\\
\mathsf{PSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{DSPACE}(p)\\
\mathsf{NPSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{NSPACE}(p)\\
\mathsf{APSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{ASPACE}(p)
\end{aligned}$$

Exponential Complexity Classes

Let $\mathcal{P}$ be the set of polynomials $p : \mathbb{N}_0 \to \mathbb{N}_0$.

Definition: EXP, NEXP, . . .

$$\begin{aligned}
\mathsf{EXP} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{DTIME}(2^p)\\
\mathsf{NEXP} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{NTIME}(2^p)\\
\mathsf{AEXP} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{ATIME}(2^p)\\
\mathsf{EXPSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{DSPACE}(2^p)\\
\mathsf{NEXPSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{NSPACE}(2^p)\\
\mathsf{AEXPSPACE} &= \textstyle\bigcup_{p\in\mathcal{P}} \mathsf{ASPACE}(2^p)
\end{aligned}$$

Doubly Exponential Complexity Classes

Let $\mathcal{P}$ be the set of polynomials $p : \mathbb{N}_0 \to \mathbb{N}_0$.

Definition: 2-EXP, ...

2-EXP $= \bigcup_{p \in \mathcal{P}} \mathsf{DSPACE}(2^{2^p})$

...

Standard Complexity Classes Relationships

Theorem

$$\begin{array}{rcl}
\mathsf{P} \subseteq & \mathsf{NP} & \subseteq \mathsf{AP} \\
\mathsf{PSPACE} \subseteq & \mathsf{NPSPACE} & \subseteq \mathsf{APSPACE} \\
\mathsf{EXP} \subseteq & \mathsf{NEXP} & \subseteq \mathsf{AEXP} \\
\mathsf{EXPSPACE} \subseteq & \mathsf{NEXPSPACE} & \subseteq \mathsf{AEXPSPACE} \\
\text{2-EXP} \subseteq & \ldots &
\end{array}$$

The Power of Nondeterministic Space

Theorem (Savitch 1970)

NSPACE$(f) \subseteq$ DSPACE(f^2), and thus:

$$
\begin{aligned}
\text{PSPACE} &= \text{NPSPACE} \\
\text{EXPSPACE} &= \text{NEXPSPACE}
\end{aligned}
$$

The Power of Alternation

Theorem (Chandra et al. 1981)

$$
\begin{aligned}
\text{AP} &= \text{PSPACE} \\
\text{APSPACE} &= \text{EXP} \\
\text{AEXP} &= \text{EXPSPACE} \\
\text{AEXPSPACE} &= \text{2-EXP}
\end{aligned}
$$

Planning and Complexity
Part 2: Classical Planning

Malte Helmert

June 7th, 2006

Outline

What We Consider

We begin with a simple planning formalism, variable-free PDDL2.1 level 1:

- deterministic
- fully observable
- grounded
- no numbers (level 2)
- no discretized durative actions (level 3)
- no continuous durative actions (level 4)
- no axioms or timed initial literals (PDDL2.2)
- no trajectory constraints or preferences (PDDL3)

Operators

Let V be a set of propositional variables.

Definition: Operator

An operator for V is a pair $\langle \chi, e \rangle$ of precondition χ and effect e, where

- χ is a propositional formula over V
- e is an effect, which is either
  - a simple add effect v, where $v \in V$,
  - a simple delete effect $\neg v$, where $v \in V$,
  - a conditional effect $\varphi \rhd e'$, where φ is a propositional formula over V and e' is an effect, or
  - a conjunctive effect $e' \wedge e''$, where e' and e'' are effects

Planning Tasks

Definition: Planning task

A planning task is a 4-tuple $\langle V, s_0, O, \chi_\star \rangle$, where

- V is a finite set of propositional state variables,
  - truth assignments to V are called states
- s_0 is the initial state,
- O is a finite set of operators for V,
- $\chi_\star$ is the goal, a propositional formula over V

Add Sets and Delete Sets

Definition: Add set, delete set

Let e be an effect and s be a state.
Define the add set $e^+(s)$ and delete set $e^-(s)$:

- simple add effect $e = v$:
  - $e^+(s) = \{v\}$
  - $e^-(s) = \emptyset$
- simple delete effect $e = \neg v$:
  - $e^+(s) = \emptyset$
  - $e^-(s) = \{v\}$

Add Sets and Delete Sets (continued)

Definition: Add set, delete set (ctd.)

Let e be an effect and s be a state.
Define the add set $e^+(s)$ and delete set $e^-(s)$:

- conditional effect $e = (\varphi \rhd e')$:
  - $e^+(s) = \begin{cases} e'^+(s) & \text{if } s \models \varphi \\ \emptyset & \text{if } s \not\models \varphi \end{cases}$
  - $e^-(s) = \begin{cases} e'^-(s) & \text{if } s \models \varphi \\ \emptyset & \text{if } s \not\models \varphi \end{cases}$
- conjunctive effect $e = (e' \wedge e'')$:
  - $e^+(s) = e'^+(s) \cup e''^+(s)$
  - $e^-(s) = e'^-(s) \cup e''^-(s)$

Operator Semantics

Definition: Applying operators

Operator $o = \langle \chi, e \rangle$ is applicable in state s iff $s \models \chi$.
The result of applying operator o (or effect e) in s, written as $o(s)$ (or $e(s)$), is the state s' with:

$$s'(v) = \begin{cases} \mathbf{T} & \text{if } v \in e^+(s) \\ \mathbf{F} & \text{if } v \in e^-(s) \text{ and } v \notin e^+(s) \\ s(v) & \text{otherwise} \end{cases}$$

Plans

Let $\Pi = \langle V, s_0, O, \chi_\star \rangle$ be a planning task.

Definition: Plan

A plan for Π is a sequence of operators $\pi = o_1 \dots o_n \in O^*$ such that:

- For all $i \in \{1, \dots, n\}$, o_i is applicable in s_{i-1}, where $s_i = o_i(s_{i-1})$
- $s_n \models \chi_\star$

The Planning Problem

PlanEx (Plan Existence)

Given:	Planning task Π
Question:	Is there a plan for Π?

PlanLen (Bounded Plan Existence)

Given:	Planning task Π, bound $K \in \mathbb{N}_0$
Question:	Is there a plan for Π of length at most K?

Outline

1. Classical Planning
2. Complexity Results
   - PlanEx vs. PlanLen
   - Membership in PSPACE
   - Hardness for PSPACE
3. First-Order Tasks

Plan Existence vs. Bounded Plan Existence

PlanEx $\leq_p$ PlanLen

A planning task with n state variables has a plan
iff it has a plan of length at most $2^n - 1$.
$\rightsquigarrow$ polynomial reduction

Membership in PSPACE

PlanLen $\in$ PSPACE

Show PlanLen $\in$ NPSPACE and use Savitch's theorem.
Nondeterministic algorithm:

def plan($\langle V, s_0, O, \chi_\star \rangle$, K):
  $s := s_0$
  $k := K$
  repeat until $s \models \chi_\star$:
    guess $o \in O$
    reject if o not applicable in s
    set $s := o(s)$
    reject if $k = 0$
    set $k := k - 1$
  accept

Hardness for PSPACE

Idea: generic reduction

- For a fixed polynomial p, given DTM M and input w, generate planning task which is solvable iff M accepts w in space $p(|w|)$
- For simplicity, restrict to TMs which never move to the left of the initial head position (no loss of generality)

Reduction: State Variables

Let p be the space bound polynomial.

Given DTM $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$,
define relevant tape positions $I = \{1, \dots, p(n)\}$.

State variables

- state_q for all $q \in Q$
- head_i for all $i \in I \cup \{0, p(n) + 1\}$
- $\mathsf{content}_{i,a}$ for all $i \in I$, $a \in \Sigma_\Box$

Reduction: Initial State

Let p be the space bound polynomial.

Given DTM $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$,
define relevant tape positions $I = \{1, \dots, p(n)\}$.

Initial state

Initially true:

- state_{q_0}
- head_1
- $\mathsf{content}_{i,w_i}$ for all $i \in \{1, \dots, n\}$
- $\mathsf{content}_{i,\Box}$ for all $i \in I \setminus \{1, \dots, n\}$

Initially false:

- all others

Reduction: Operators

Let p be the space bound polynomial.

Given DTM $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$,
define relevant tape positions $I = \{1, \dots, p(n)\}$.

Operators

One operator for each transition rule $\delta(q, a) = (q', a', \Delta)$ and each cell position $i \in I$:

- precondition: $\mathsf{state}_q \wedge \mathsf{head}_i \wedge \mathsf{content}_{i,a}$
- effect: $\neg\mathsf{state}_q \wedge \neg\mathsf{head}_i \wedge \neg\mathsf{content}_{i,a}$
  $\wedge\, \mathsf{state}_{q'} \wedge \mathsf{head}_{i+\Delta} \wedge \mathsf{content}_{i,a'}$

Reduction: Goal

Let p be the space bound polynomial.

Given DTM $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$,
define relevant tape positions $I = \{1, \dots, p(n)\}$.

Goal

$\bigvee_{q \in Q_Y} \mathsf{state}_q$

Outline

First-Order Tasks

- we considered
  propositional state variables (0-ary predicates) and
  grounded operators (0-ary schematic operators)
- reasonable: most planning algorithms
  work on grounded representations
- predicate arity is typically small (a constant?)

How do the complexity results change if we introduce first-order predicates and schematic operators?

$\rightsquigarrow$ formalization omitted

Membership in EXPSPACE

$\textsc{PlanEx}, \textsc{PlanLen} \in$ EXPSPACE

- input size n
- $\rightsquigarrow$ at most 2^n grounded state variables
- $\rightsquigarrow$ at most 2^n grounded operators
- can ground the task in exponential time, then use the earlier PSPACE algorithms

Hardness for EXPSPACE

Idea: Adapt the earlier reduction from $\textsc{PlanEx}$ to encode Turing Machine contents more succinctly.
Assume relevant tape positions are now $I = \{1, \dots, 2^n\}$.
We need to encode the computation as a planning task in polynomial time!

Objects

$0, 1$

Predicates

- $\text{state}_q()$ for all $q \in Q$
- $\text{head}(?b_1, \dots, ?b_n)$
- $\text{content}_a(?b_1, \dots, ?b_n)$

Reduction: Example Operator

Operator example

Schematic operator for transition rule $\delta(q, a) = (q', a', +1)$

- parameters: $?b_1, \dots, ?b_n$
- precondition:
  state_q
  $\wedge\ \text{head}(?b_1, \dots, ?b_n)$
  $\wedge\ \text{content}_a(?b_1, \dots, ?b_n)$
- effect:
  $\neg\text{state}_q$
  $\wedge\ \neg\text{head}(?b_1, \dots, ?b_n)$
  $\wedge\ \neg\text{content}_a(?b_1, \dots, ?b_n)$
  $\wedge\ \text{state}_{q'}$
  $\wedge$ advance-head
  $\wedge\ \text{content}'_a(?b_1, \dots, ?b_n)$

Reduction: Example Operator (continued)

Operator example (ctd.)

$$
\begin{aligned}
\text{advance-head} = {} & ((?b_n = 0) \\
& \quad \rhd \text{head}(?b_1, \dots, ?b_{n-1}, 1)) \\
\wedge\ & ((?b_{n-1} = 0 \wedge ?b_n = 1) \\
& \quad \rhd \text{head}(?b_1, \dots, ?b_{n-2}, 1, 0)) \\
\wedge\ & ((?b_{n-2} = 0 \wedge ?b_{n-1} = 1 \wedge ?b_n = 1) \\
& \quad \rhd \text{head}(?b_1, \dots, ?b_{n-3}, 1, 0, 0)) \\
\wedge\ & \dots \\
\wedge\ & ((?b_1 = 0 \wedge ?b_2 = 1 \wedge \cdots \wedge ?b_n = 1) \\
& \quad \rhd \text{head}(1, 0, \dots, 0))
\end{aligned}
$$

Plan Existence vs. Bounded Plan Existence

- Our earlier reduction from PLANEX to PLANLEN no longer works: the shortest plan can have length doubly exponentially in the input size, so that the bound cannot be written down in polynomial time
- Indeed, PLANLEN is actually easier than PLANEX for this planning formalism (NEXP-complete).

Planning and Complexity
Part 3: Conditional Planning

Malte Helmert

June 7th, 2006

Outline

1. Conditional Planning
   - Overview
   - Planning Tasks
   - Plans
   - The Planning Problem
   - Special Cases
2. Full Observability
3. Deterministic Operators
4. General Case

Overview

Extend planning model by

- nondeterminism and
- restricted observability

First consider each in isolation, then combination

Nondeterministic Operators

Let V be a set of propositional variables.

Definition: Nondeterministic operator

An operator for V is a pair $\langle \chi, e \rangle$
of precondition χ and effect e, where

- χ is a propositional formula over V
- e is an effect, which is either
  - a simple add effect v, where $v \in V$,
  - a simple delete effect $\neg v$, where $v \in V$,
  - a conditional effect $\varphi \rhd e'$, where φ is a propositional formula over V and e' is an effect,
  - a conjunctive effect $e' \wedge e''$, where e' and e'' are effects, or
  - a choice effect $e'|e''$, where e' and e'' are effects

Planning Tasks

Definition: Planning task

A planning task is a 5-tuple $\langle V, V_\text{o}, \chi_0, O, \chi_\star \rangle$, where

- V is a finite set of propositional state variables
  - truth assignments to V are called states
  - sets of states are called belief states
- $V_\text{o} \subseteq V$ is the set of observable state variables
- χ_0 is the initial states formula
- O is a finite set of nondeterministic operators for V
- $\chi_\star$ is the goal, a propositional formula over V

Possibilities

Definition: Possibilities

Let e be a nondeterministic effect.
Define the set of possibilities $\mathit{poss}(e)$:

- simple add or delete effect e:
  $\mathit{poss}(e) = \{e\}$
- conditional effect $e = (\varphi \rhd e')$:
  $\mathit{poss}(e) = \{\ \varphi \rhd e'_\text{p} \mid e'_\text{p} \in \mathit{poss}(e')\ \}$
- conjunctive effect $e = (e' \wedge e'')$:
  $\mathit{poss}(e) = \{\ e'_\text{p} \wedge e''_\text{p} \mid e'_\text{p} \in \mathit{poss}(e'), e''_\text{p} \in \mathit{poss}(e'')\ \}$
- choice effect $e = e' | e''$:
  $\mathit{poss}(e) = \mathit{poss}(e') \cup \mathit{poss}(e'')$

Operator Semantics

Definition: Applying operators

Operator $o = \langle \chi, e \rangle$ is applicable in belief state B iff $s \models \chi$ for all $s \in B$.
The result of applying o in B, written as $o(B)$, is defined as:
$o(B) = \{ e_{\mathsf{p}}(s) \mid s \in B, e_{\mathsf{p}} \in \mathit{poss}(e) \}$

Observations

Let $\Pi = \langle V, V_{\mathsf{o}}, \chi_0, O, \chi_\star \rangle$ be a planning task.

Definition: Observations

An observation φ for Π is a propositional formula over the observable state variables V_{o}.
The positive result of applying φ to a belief state B is the belief state $\varphi^+(B) = \{ s \in B \mid s \models \varphi \}$.
The negative result of applying φ to a belief state B is the belief state $\varphi^-(B) = \{ s \in B \mid s \not\models \varphi \}$.

Conditional Plans

Nondeterminism:

- must extend notion of plans beyond action sequences
- need strategies or policies, or even programs

Different kinds of reachability:
weak, strong cyclic, strong plans

- weak plans are fairly uninteresting
- we consider strong (acyclic) plans
- results extend to cyclic plans

Plans

Let $\Pi = \langle V, V_\circ, \chi_0, O, \chi_\star \rangle$ be a planning task.

Definition: Plan

A plan for Π is a finite tree of

- operator nodes for some operator o
- observation nodes for some observation φ
- goal nodes

All nodes have an associated belief state B.
Operator and observation nodes are internal nodes,
goal nodes are leaves.
...

Plans (continued)

Let $\Pi = \langle V, V_{\mathsf{o}}, \chi_0, O, \chi_\star \rangle$ be a planning task.

Definition: Plan (ctd.)

The plan must satisfy the following properties:

- root node:
  belief state contains a state s iff $s \models \chi_0$
- operator nodes:
  operator o is applicable in B,
  exactly one child, with belief state $o(B)$
- observation nodes:
  exactly two children, with belief states $\varphi^+(B)$, $\varphi^-(B)$
- goal nodes:
  belief state contains state s only if $s \models \chi_\star$

Cyclic Plans

Side Remark

We could adjust the definition to cyclic plans as follows:

- instead of trees, allow general directed graphs
- instead of root, have dedicated initial node
- require that each node can reach some goal node

The Planning Problem

PLANEX (Plan Existence)

GIVEN:	Planning task Π
QUESTION:	Is there a plan for Π?

- we do not consider bounded plan existence
- notions of plan size become more complicated
- important issue: plan representation

Special Cases

- fully observable case ($V_\mathrm{o} = V$)
  $\rightsquigarrow$ can assume that all belief sets are singletons
  (always observe until this is true)
- deterministic operator case (no choice effects)
  $\rightsquigarrow$ do not need observations
  $\rightsquigarrow$ can assume that plans are action sequences

Outline

1. Conditional Planning
2. Full Observability
   - Membership in EXP
   - Hardness for EXP
3. Deterministic Operators
4. General Case

Membership in EXP

PLANEX $\in$ EXP

Backward induction

def plan($\langle V, \chi_0, O, \chi_\star \rangle$):
  Let S be the set of states.
  solved $:= \{ s \in S \mid s \models \chi_\star \}$
  repeat until fixpoint:
    for each $s \in S$, $o \in O$:
      if o applicable in $\{s\}$ and $o(\{s\}) \subseteq$ *solved*:
        solved $:=$ *solved* $\cup \{s\}$
  accept iff $s \in$ *solved* for all states s with $s \models \chi_0$

Hardness for EXP

Idea:

- adapt hardness proof for classical case to alternating Turing Machines
- existential states
  $\rightsquigarrow$ separate operators
- universal states
  $\rightsquigarrow$ operators with nondeterministic effects

Reduction: State Variables

Let p be the space bound polynomial.

Given ATM $\langle \Sigma, \square, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$, define relevant tape positions $I = \{1, \dots, p(n)\}$.

State variables

- state_q for all $q \in Q$
- head_i for all $i \in I \cup \{0, p(n) + 1\}$
- $\mathsf{content}_{i,a}$ for all $i \in I$, $a \in \Sigma_\square$

Reduction: Initial State

Initial state

Initially true:

- state_{q0}
- head_1
- $\text{content}_{i,w_i}$ for all $i \in \{1, \dots, n\}$
- $\text{content}_{i,\square}$ for all $i \in I \setminus \{1, \dots, n\}$

Initially false:

- all others

Reduction: Operators

Operators

For $q, q' \in Q$, $a, a' \in \Sigma_\square$, $\Delta \in \{-1, +1\}$, $i \in I$, define

- $\text{pre}_{q,a,i} = \text{state}_q \wedge \text{head}_i \wedge \text{content}_{i,a}$
- $\text{eff}_{q,a,q',a',\Delta,i} = \neg\text{state}_q \wedge \neg\text{head}_i \wedge \neg\text{content}_{i,a}$
  $\wedge\ \text{state}_{q'} \wedge \text{head}_{i+\Delta} \wedge \text{content}_{i,a'}$

Reduction: Operators (continued)

Operators (ctd.)

For existential states $q \in Q_\exists$, $a \in \Sigma_\square$, $i \in I$:
Let (q'_j, a'_j, Δ_j) $(j \in \{1, \dots, k\})$ be those triples with $((q, a), (q'_j, a'_j, \Delta_j)) \in \delta$.

For each $j \in \{1, \dots, k\}$, one operator:

- precondition: $\mathsf{pre}_{q,a,i}$
- effect: $\mathsf{eff}_{q,a,q'_j,a'_j,\Delta_j,i}$

Reduction: Operators (continued)

Operators (ctd.)

For universal states $q \in Q_\forall$, $a \in \Sigma_\square$, $i \in I$:
Let (q'_j, a'_j, Δ_j) $(j \in \{1, \dots, k\})$ be those triples with $((q, a), (q'_j, a'_j, \Delta_j)) \in \delta$.

One operator:

- precondition: $\mathsf{pre}_{q,a,i}$
- effect: $\mathsf{eff}_{q,a,q'_1,a'_1,\Delta_1,i} | \dots | \mathsf{eff}_{q,a,q'_k,a'_k,\Delta_k,i}$

Reduction: Goal

Goal

$\bigvee_{q \in Q_Y} \mathsf{state}_q$

Outline

Membership in EXPSPACE

PlanEx $\in$ EXPSPACE

Generate a classical propositional planning task which has one state variable for each state of the input task.

- states of the generated planning task correspond to belief states of the input task
- operators, initial states, goal easy to convert

$\rightsquigarrow$ exponential-time reduction to a problem in PSPACE
$\rightsquigarrow$ EXPSPACE algorithm

Hardness for EXPSPACE

Idea:

- generic reduction for DTMs with exponential space
- TM states and tape head position easily representable with polynomially many state variables

Problem:

- must encode exponentially many tape cell contents in polynomially many state variables

Hardness for EXPSPACE (continued)

The trick:

- only keep track of the contents one tape cell
  $\rightsquigarrow$ watched tape cell
- which tape cell is watched is unobservable
- $\rightsquigarrow$ plan must work correctly for all possible choices
- $\rightsquigarrow$ plan must remain faithful to the TM computation

Reduction: State Variables

Let p be a polynomial such that 2^p is a space bound.

Given DTM $\langle \Sigma, \Box, Q, q_0, l, \delta \rangle$ and input $w_1 \dots w_n$,
define relevant tape positions $I = \{1, \dots, 2^p(n)\}$.

State variables

Convention:
Use bars to denote vectors of $p(n)$ state variables encoding a number in $\{1, \dots, 2^p(n)\}$.

- state_q for all $q \in Q$
- $\overline{\mathsf{head}}$
- $\mathsf{content}_a$ for all $a \in \Sigma_\Box$
- $\overline{\mathsf{watched}}$

Reduction: Initial State Formula

Initial state formula

$$
\begin{aligned}
\chi_0 = \mathsf{state}_{q_0} & \wedge \bigwedge_{q \in Q \setminus \{q_0\}} \neg \mathsf{state}_q \\
& \wedge \overline{\mathsf{head}} = 1 \\
& \wedge \bigwedge_{i=1}^{n} ((\overline{\mathsf{watched}} = i) \rightarrow \mathsf{content}_{w_i}) \\
& \wedge (\overline{\mathsf{watched}} > n) \rightarrow \mathsf{content}_{\square} \\
& \wedge \bigwedge_{a \in \Sigma_{\square}} \bigwedge_{a' \in \Sigma_{\square} \setminus \{a\}} \neg(\mathsf{content}_a \wedge \mathsf{content}_{a'})
\end{aligned}
$$

Note: watched tape cell unspecified

Reduction: Operators

Operators

One operator for each transition rule $\delta(q, a) = (q', a', \Delta)$:

- precondition:
  state_q
  $\wedge ((\overline{\mathsf{head}} = \overline{\mathsf{watched}}) \rightarrow \mathsf{content}_a)$
  If $\Delta = -1$, conjoin with $\overline{\mathsf{head}} > 1$.
  If $\Delta = +1$, conjoin with $\overline{\mathsf{head} < 2^{p(n)}}$.
- effect:
  $\neg\mathsf{state}_q$
  $\wedge\, \mathsf{state}_{q'}$
  $\wedge\, (\overline{\mathsf{head}} := \overline{\mathsf{head}} + \Delta)$
  $\wedge\, ((\overline{\mathsf{head}} = \overline{\mathsf{watched}}) \rightarrow (\neg\mathsf{content}_a \wedge \mathsf{content}_{a'}))$

Reduction: Goal

Goal

$\bigvee_{q \in Q_Y} \mathsf{state}_q$

Outline

Membership in 2-EXP

PLANEX $\in$ 2-EXP

Explicitly construct the transition graph for the set of belief states, then solve by backward induction.
(Translate observations into operators.)

Hardness for 2-EXP

PLANEX is 2-EXP-hard (Rintanen 2004)

Combine the techniques of the previous two proofs:

- Consider alternating Turing Machine with exponential space
- Use (unobservable) watched tape cell to reduce planning task description from exponential to polynomial size
- $\rightsquigarrow$ result follows with AEXPSPACE = 2-EXP

Planning and Complexity
Part 4: Numeric Planning

Malte Helmert

June 7th, 2006

Outline

Infinite State Spaces

We now introduce numbers.

- $\rightsquigarrow$ infinite state spaces
- $\rightsquigarrow$ decidability issues

Where Numbers Appear

Base formalism:

- STRIPS subset of PDDL2.1 level 2

- Numerical state variables
- Numerical preconditions
- Numerical goal conditions
- Numerical effects

$\rightsquigarrow$ How does the type of numerical conditions and effects affect the hardness of the problem?

Planning Formalisms

Definition: Planning formalism

A planning formalism is represented by a triple $\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$, where $\mathcal{G}, \mathcal{P}, \mathcal{E} \subseteq \bigcup_{k \in \mathbb{N}} (\mathbb{Q}^k \to \mathbb{Q})$.

- $\mathcal{G}$: functions in goal conditions
- $\mathcal{P}$: functions in operator preconditions
- $\mathcal{E}$: functions in operator effects

States

From now, V_P and V_N are disjoint finite sets of variables, and $\mathcal{C}, \mathcal{G}, \mathcal{P}, \mathcal{E} \subseteq \bigcup_{k \in \mathbb{N}} (\mathbb{Q}^k \to \mathbb{Q})$.

Definition: State

A state over $\langle V_\mathsf{P}, V_\mathsf{N} \rangle$ is a pair

$$\langle \alpha, \beta \rangle \in (V_\mathsf{P} \to \{\mathbf{T}, \mathbf{F}\}) \times (V_\mathsf{N} \to \mathbb{Q}).$$

Conditions

Definition: Condition

A condition over $\langle V_{\mathsf{P}}, V_{\mathsf{N}}, \mathcal{C}\rangle$ is either

- a propositional condition:
  v or $\neg v$ $(v \in V_{\mathsf{P}})$, or
- a numerical condition:
  $f(v_1, \dots, v_k)$ **relop** 0
  for $k \in \mathbb{N}, f \in \mathcal{C}$ of arity k, $v_1, \dots, v_k \in V_{\mathsf{N}}$,
  relop $\in \{=, \neq, <, \leq, \geq, >\}$

Effects

Definition: Effect

An effect over $\langle V_{\mathsf{P}}, V_{\mathsf{N}}, \mathcal{E}\rangle$ is either

- a propositional effect:
  v or $\neg v$ $(v \in V_{\mathsf{P}})$, or
- a numerical effect:
  $v_0 := f(v_0, v_1, \dots, v_k)$
  for $k \in \mathbb{N}_0$, $f \in \mathcal{E}$ of arity $k+1$, $v_0, v_1, \dots, v_k \in V_{\mathsf{N}}$

Operators

Definition: Operator

An operator over $\langle V_\mathsf{P}, V_\mathsf{N}, \mathcal{C}, \mathcal{E}\rangle$ is a pair $\langle p, e\rangle$, where

- p is a finite set of conditions over $\langle V_\mathsf{P}, V_\mathsf{N}, \mathcal{C}\rangle$
- e is a finite set of effects over $\langle V_\mathsf{P}, V_\mathsf{N}, \mathcal{E}\rangle$

Operator Semantics

Definition: Applicable operators

Operator $\langle p, e\rangle$ is applicable in state $\langle \alpha, \beta\rangle$ iff

- for all propositional conditions $l \in p$:
  $\alpha \models l$
- for all numerical conditions $f(v_1, \dots, v_k)$ $\mathbf{relop}$ $0 \in p$:
  $f(\beta(v_1), \dots, \beta(v_k))$ $\mathbf{relop}$ 0

Operator Semantics (continued)

Definition: Applying operators

The result of applying operator $\langle p, e\rangle$ to state $\langle \alpha, \beta\rangle$ is the state $\langle \alpha', \beta'\rangle$ with:

$$\alpha'(v) = \begin{cases} \mathbf{T} & \text{if } v \in e \\ \mathbf{F} & \text{if } \neg v \in e \text{ and } v \notin e \\ \alpha(v) & \text{otherwise} \end{cases}$$

$$\beta'(v_0) = \begin{cases} f(\beta(v_0), \dots, \beta(v_k)) & \text{if } v_0 := f(v_0, \dots, v_k) \in e \\ \beta(v_0) & \text{otherwise} \end{cases}$$

Planning Tasks

Definition: Planning task

A planning task of planning formalism $\langle \mathcal{G}, \mathcal{P}, \mathcal{E}\rangle$ is a 5-tuple $\langle V_\mathsf{P}, V_\mathsf{N}, s_0, G, O\rangle$ with:

- V_P finite set of propositional state variables
- V_N finite set of numerical state variables (disjoint from V_P)
- s_0 state over $\langle V_\mathsf{P}, V_\mathsf{N}\rangle$ (initial state)
- G finite set of conditions over $\langle V_\mathsf{P}, V_\mathsf{N}, \mathcal{G}\rangle$ (goal)
- O finite set of operators over $\langle V_\mathsf{P}, V_\mathsf{N}, \mathcal{P}, \mathcal{E}\rangle$

$\rightsquigarrow$ definition of plans omitted

Outline

1. Numeric Planning
2. Problem Hierarchy
   - The Planning Problem
   - Condition Types
   - Effect Types
   - Problem Family
3. Undecidable Cases
4. Decidable Cases
5. Conclusion

Plan Existence

Let $\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$ be a planning formalism.

PlanEx-$\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$ (Plan Existence)

Given:	Planning task Π of formalism $\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$
Question:	Is there a plan for Π?

Possible Values for $\mathcal{G}$ and $\mathcal{P}$

- $\mathcal{C}_\emptyset = \emptyset$:
  no numerical conditions
- $\mathcal{C}_0 = \{x \mapsto x\}$:
  compare to 0
- $\mathcal{C}_c = \{\ x \mapsto x - c \mid c \in \mathbb{Q}\ \}$:
  compare to constants
- $\mathcal{C}_= = \{\ (x_1, x_2) \mapsto x_1 - x_2\ \}$:
  compare other numerical state variable
- $\mathcal{C}_p = \mathbb{Q}[x]$:
  compare polynomial of state variable to 0
- $\mathcal{C}_{p+} = \mathbb{Q}[x_1, x_2, x_3, \ldots]$:
  compare polynomial of multiple state variables to 0

Possible Values for $\mathcal{G}$ and $\mathcal{P}$: Hierarchy

$$\begin{tikzcd}
\mathcal{C}_\emptyset \arrow[r] \arrow[drr] & \mathcal{C}_0 \arrow[r] & \mathcal{C}_c \arrow[r] & \mathcal{C}_p \arrow[dr] & \\
 & & \mathcal{C}_= \arrow[rr] & & \mathcal{C}_{p+}
\end{tikzcd}$$

Possible Values for $\mathcal{E}$

- $\mathcal{E}_\emptyset = \emptyset$:
  no numerical effects
- $\mathcal{E}_{+1} = \{\ x \mapsto x + c \mid c \in \{+1\}\ \}$:
  increase by 1
- $\mathcal{E}_{\pm 1} = \{\ x \mapsto x + c \mid c \in \{-1, +1\}\ \}$:
  increase or decrease by 1
- $\mathcal{E}_{+c} = \{\ x \mapsto x + c \mid c \in \mathbb{Q}_+\ \}$:
  increase by constants
- $\mathcal{E}_{\pm c} = \{\ x \mapsto x + c \mid c \in \mathbb{Q}\ \}$:
  increase or decrease by constants

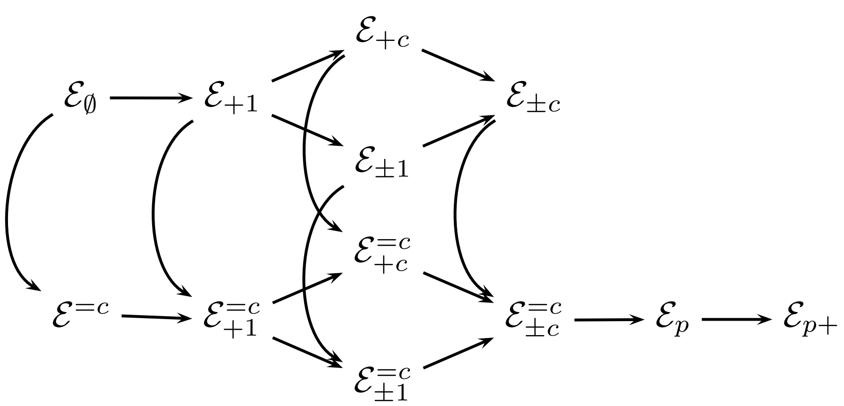
Possible Values for $\mathcal{E}$ (continued)

- $\mathcal{E}^{=c} = \{\ x \mapsto c \mid c \in \mathbb{Q}\ \}$:
  assign constant
- $\mathcal{E}^{=c}_{+1} = \mathcal{E}^{=c} \cup \mathcal{E}_{+1}$:
  increase by 1 or assign constant
- $\mathcal{E}^{=c}_{\pm 1} = \mathcal{E}^{=c} \cup \mathcal{E}_{\pm 1}$:
  increase or decrease by 1 or assign constants
- $\mathcal{E}^{=c}_{+c} = \mathcal{E}^{=c} \cup \mathcal{E}_{+c}$:
  increase by constants or assign constant
- $\mathcal{E}^{=c}_{\pm c} = \mathcal{E}^{=c} \cup \mathcal{E}_{\pm c}$:
  increase or decrease by constants, or assign constants

Possible Values for $\mathcal{E}$ (continued)

- $\mathcal{E}_p = \mathbb{Q}[x]$:
  assign polynomial of old value
- $\mathcal{E}_{p+} = \mathbb{Q}[x_1, x_2, x_3, \ldots]$:
  assign polynomial of old values of multiple state variables

Possible Values for $\mathcal{E}$: Hierarchy



How Many Questions to Answer?

Investigate decidability status of $\textsc{PlanEx}$-$\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$ for...

- 6 values of $\mathcal{G}$
- 6 values of $\mathcal{P}$
- 12 values of $\mathcal{E}$

$\rightsquigarrow$ 432 combinations (not all interesting)

Outline

1. Numeric Planning
2. Problem Hierarchy
3. Undecidable Cases
   - Diophantine Equations
   - PCP
   - Abacus Programs
4. Decidable Cases
5. Conclusion

Multi-Variable Polynomials in Goals

Theorem: $\textsc{PlanEx}$-$\langle \mathcal{C}_{p+}, \mathcal{C}_{\emptyset}, \mathcal{E}_{+1} \rangle$ is undecidable

Proof idea.

$\textsc{Diophant}_{\mathbb{N}_0}$: Given $k \in \mathbb{N}$ and $p \in \mathbb{Q}[x_1, \dots, x_k]$, does p have a solution in $\mathbb{N}_0^k$?

Map to planning task:

- State variables: numerical variables $x_1, \dots, x_k$
- Initial state: all set to 0
- Goal: $p(x_1, \dots, x_k) = 0$
- Operators: one operator [EFF: $x_i := x_i + 1$] for each $i \in \{1, \dots, k\}$

Comparing Variables in Goals Polynomials in Effects

Theorem: $\textsc{PlanEx}$-$\langle \mathcal{C}_{=}, \mathcal{C}_{\emptyset}, \mathcal{E}_{p} \rangle$ is undecidable

Proof idea.

MPCP7: Given word pairs $(a_1, b_1), \dots, (a_7, b_7)$ in $\{1, 2\}^*$, is there a sequence $i_1, \dots, i_M \in \{1, \dots, 7\}^+$ with $i_1 = 1$ and $a_{i_1} a_{i_2} \dots a_{i_M} = b_{i_1} b_{i_2} \dots b_{i_M}$?

Map to planning task:

- State variables: numerical variables a, b
- Initial state: a set to $\#a_1$, b set to $\#b_1$
- Goal: $a = b$
- Operators: one operator [EFF: $a := 10^{|a_i|} a + \#a_i$; $b := 10^{|b_i|} b + \#b_i$] for each $i \in \{1, \dots, 7\}$

Comparing to Zero in Goals
Polynomials in Effects

Theorem: $\textsc{PlanEx}\text{-}\langle \mathcal{C}_0, \mathcal{C}_\emptyset, \mathcal{E}_p \rangle$ is undecidable

Proof idea.

very similar reduction from $\mathrm{MPCP7}$

Abacus Programs

Definition: Abacus program

An abacus program is a 5-tuple $\langle V, L, l_0, l_H, P \rangle$:

- V finite set of variables or registers
- L finite set of labels
- start label $l_0 \in L$
- halt label $l_\star \in L$
- program P: mapping of labels to
  - increment statements
    INC $v; \rightarrow l'$ for $v \in V$, $l' \in L$, or
  - conditional decrement statements
    DEC $v; \rightarrow l_=, l_>$ for $v \in V$, $l_=, l_> \in L$

$\rightsquigarrow$ semantics omitted

$\rightsquigarrow$ abacus program formalism is Turing-complete

Comparing to Zero in Preconditions
Add/Subtract 1 in Effects

Theorem: $\text{PLANEX-}\langle \mathcal{C}_\emptyset, \mathcal{C}_0, \mathcal{E}_{\pm 1} \rangle$ is undecidable

Proof idea.

Map abacus program $\langle V, L, l_0, l_\star, P \rangle$ to planning task:

- State variables: propositional: L, numerical: V
- Initial state: l_0 set to **T**, other labels set to **F**; numerical variables set to 0
- Goal condition: $l_\star$
- Operators:
  for $P(l) = \text{INC } v; \rightarrow l'$:
  - [PRE: l; EFF: $\neg l$; l'; $v := v + 1$]

  for $P(l) = \text{DEC } v; \rightarrow l_=, l_>$:
  - [PRE: l; $v = 0$; EFF: $\neg l$; $l_=$]
  - [PRE: l; $v > 0$; EFF: $\neg l$; $l_>$; $v := v - 1$]

Comparing Variables in Preconditions
Adding 1 in Effects

Theorem: $\text{PLANEX-}\langle \mathcal{C}_\emptyset, \mathcal{C}_=, \mathcal{E}_{+1} \rangle$ is undecidable

Proof idea.
very similar reduction from halting problem
for abacus programs

Outline

Trivial Results

Theorem: $\textsc{PlanEx}\text{-}\langle \mathcal{C}_\emptyset, \mathcal{C}_\emptyset, \mathcal{E}_{p+} \rangle$ is decidable

Proof idea.
ignore numerical state variables

Theorem: $\textsc{PlanEx}\text{-}\langle \mathcal{C}_{p+}, \mathcal{C}_{p+}, \mathcal{E}^{=c} \rangle$ is decidable

Proof idea.
numerical variables assume a finite range of values
$\rightsquigarrow$ compile away

Scalable Function Sets

Definition: scalable function sets

A set of rational functions $\mathcal{F}$ is called scalable if for each $q \in \mathbb{Q}_+$ and for each n-ary function $f \in \mathcal{F}$ there is some function $f_{[q]} \in \mathcal{F}$ such that for all $x_1, \dots, x_n \in \mathbb{Q}$,

$$\operatorname{sgn}(f(x_1, \dots, x_n)) = \operatorname{sgn}(f_{[q]}(qx_1, \dots, qx_n)).$$

Examples:

- $\mathcal{C}_p$: $(x \mapsto p(x))_{[q]} = (x \mapsto p(\frac{x}{q}))$
- $\mathcal{C}_c$: $(x \mapsto x - c)_{[q]} = (x \mapsto x - qc)$
- $\mathcal{C}_=$: $((x_1, x_2) \mapsto x_1 - x_2)_{[q]} = ((x_1, x_2) \mapsto x_1 - x_2)$

Domain Simplification

Theorem: Domain simplification

Let $\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$ be a planning formalism such that $\mathcal{G}$ and $\mathcal{P}$ are scalable and $\mathcal{E} \in \{\mathcal{E}^{=c}, \mathcal{E}_{+c}, \mathcal{E}^{=c}_{+c}, \mathcal{E}_{\pm c}, \mathcal{E}^{=c}_{\pm c}\}$.

Then tasks of that formalism can be effectively transformed (within the same formalism) so that

- numerical effects are of the type $v := c$ or $v := v + c$ for $c \in \mathbb{Z}$
- initial values of numerical state variables are integers

Proof idea.

... [continued on next slide]

$\rightsquigarrow$ numerical state variables only assume integer values

Domain Simplification (continued)

Theorem: Domain simplification (ctd.)

Proof idea.

- find common denominator d of rationals in the task
- multiply initial values by d
- replace $v := c$ by $v := dc$
- replace $v := v + c$ by $v := v + dc$
- replace conditions on function f by conditions on $f_{[d]}$

Condition Simplification

Theorem: Condition simplification

Let $\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle$ be a planning formalism such that $\mathcal{G}$ and $\mathcal{P}$ are scalable and $\mathcal{E} \in \{\mathcal{E}^{=c}, \mathcal{E}_{+c}, \mathcal{E}^{=c}_{+c}, \mathcal{E}_{\pm c}, \mathcal{E}^{=c}_{\pm c}\}$.

Then $\textsc{PlanEx}\text{-}\langle \mathcal{G}, \mathcal{P}, \mathcal{E} \rangle \leq_{\mathsf{T}}$
$\textsc{PlanEx}\text{-}\langle (\mathcal{G} \setminus \mathcal{C}_p) \cup \mathcal{C}_c, (\mathcal{P} \setminus \mathcal{C}_p) \cup \mathcal{C}_c, \mathcal{E} \rangle$.

Proof idea.

. . . [continued on next slide]

Condition Simplification (continued)

Theorem: Condition simplification (ctd.)

Proof idea.

- apply domain simplification
- for each condition $p(v)$ **relop** 0, calculate integers l, u such that $l < x < u$ for all x satisfying $p(x) = 0$
  - replace the condition by:

$$\begin{array}{rl} & (v \leq l \wedge p(l)\ \mathbf{relop}\ 0) \\ \vee & (v = l + 1 \wedge p(l+1)\ \mathbf{relop}\ 0) \\ \vee & \ldots \\ \vee & (v = u - 1 \wedge p(u-1)\ \mathbf{relop}\ 0) \\ \vee & (v \geq u \wedge p(u)\ \mathbf{relop}\ 0) \end{array}$$

- compile away disjunctions

Precondition Elimination

Theorem: Precondition elimination

Let $\mathcal{G}$ be a scalable function set and $\mathcal{E} \in \{\mathcal{E}^{=c}, \mathcal{E}_{+c}, \mathcal{E}^{=c}_{+c}\}$.
Then $\textsc{PlanEx}\text{-}\langle \mathcal{G}, \mathcal{C}_p, \mathcal{E} \rangle \leq_{\mathsf{T}} \textsc{PlanEx}\text{-}\langle \mathcal{G}, \mathcal{C}_\emptyset, \mathcal{E} \rangle$.

Proof idea.

- ranges of numerical state variables
  fall into finitely many equivalence classes
- introduce proposition for each equivalence class

Cycle Counting

Theorem: $\textsc{PlanEx}$-$\langle \mathcal{C}_c \cup \mathcal{C}_=, \mathcal{C}_\emptyset, \mathcal{E}_{\pm c}^{=c} \rangle$ is decidable

Cycle Counting algorithm

numerical state variables only matter for the goal
$\rightsquigarrow$ focus on finite propositional part

- compute set $\Pi_\star$ of all non-looping paths from propositional initial state to a propositional goal state
- compute set Π_c of all minimal cycles in the propositional state space
- generate integer program representing valid plans:
  ... [continued on next slide]
- solve integer program

Cycle Counting (continued)

Theorem: $\textsc{PlanEx}$-$\langle \mathcal{C}_c \cup \mathcal{C}_=, \mathcal{C}_\emptyset, \mathcal{E}_{\pm c}^{=c} \rangle$ is decidable (ctd.)

Cycle Counting algorithm

generate integer program representing valid plans:

- one $\{0, 1\}$-variable for each path in $\Pi_\star$
- one $\mathbb{N}_0$-variable for each cycle in Π_c
- constraint: choose exactly one path in $\Pi_\star$
- constraint: only choose cycles incident to chosen path
- constraint: goal condition is satisfied

Combining Cycle Counting and Simplification

Corollary

PLANEX-$\mathcal{F}$ is decidable for:

1. $\mathcal{F} = \langle \mathcal{C}_p, \mathcal{C}_\emptyset, \mathcal{E}^{=c}_{\pm c} \rangle$
2. $\mathcal{F} = \langle \mathcal{C}_=, \mathcal{C}_\emptyset, \mathcal{E}^{=c}_{\pm c} \rangle$
3. $\mathcal{F} = \langle \mathcal{C}_p, \mathcal{C}_p, \mathcal{E}^{=c}_{+c} \rangle$
4. $\mathcal{F} = \langle \mathcal{C}_=, \mathcal{C}_p, \mathcal{E}^{=c}_{+c} \rangle$

Proof idea.

- for 1 and 3: condition simplification
- for 3 and 4: precondition elimination
- cycle counting

Outline

Summary of Results

type of effects	decidable iff...
$\mathcal{E}_\emptyset$, $\mathcal{E}^{=c}$	always
$\mathcal{E}_{+1}$, $\mathcal{E}^{=c}_{+1}$, $\mathcal{E}_{+c}$, $\mathcal{E}^{=c}_{+c}$	$\mathcal{G} \neq \mathcal{C}_{p+}$ and $\mathcal{P} \notin \{\mathcal{C}_=, \mathcal{C}_{p+}\}$
$\mathcal{E}_{\pm 1}$, $\mathcal{E}^{=c}_{\pm 1}$, $\mathcal{E}_{\pm c}$, $\mathcal{E}^{=c}_{\pm c}$	$\mathcal{G} \neq \mathcal{C}_{p+}$ and $\mathcal{P} = \mathcal{C}_\emptyset$
$\mathcal{E}_p$, $\mathcal{E}_{p+}$	$\mathcal{G} = \mathcal{C}_\emptyset$ and $\mathcal{P} = \mathcal{C}_\emptyset$

- all undecidability results still hold
  if $\{=, \neq\}$ are the only relational operators
- all undecidable formalisms still semi-decidable