CSC2542
Introduction to Planning

Sheila McIlraith
Department of Computer Science
University of Toronto
Summer 2014

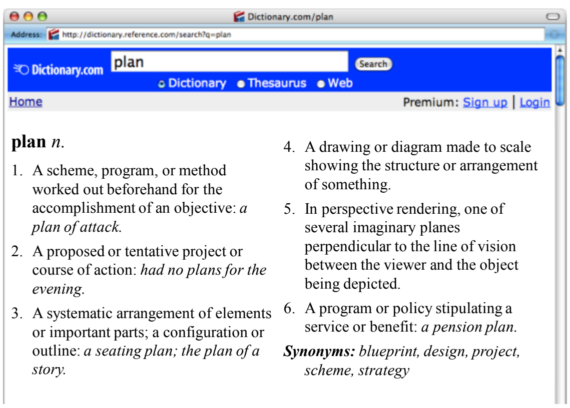
Acknowledgements

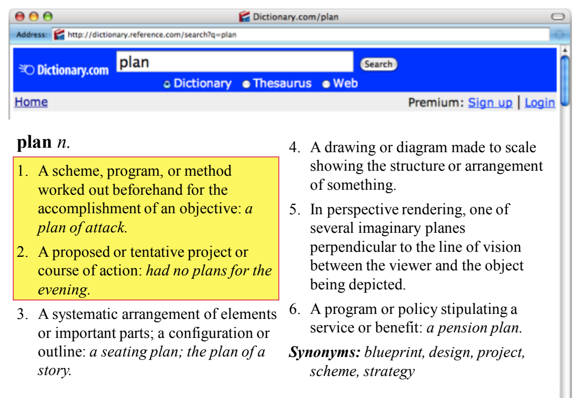
Some of the slides used in this course are modifications of Dana Nau's lecture slides for the textbook *Automated Planning,* licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License: http://creativecommons.org/licenses/by-nc-sa/2.0/

Other slides are modifications of slides developed by Malte Helmert, Bernhard Nebel, and Jussi Rintanen.

I have also used some material prepared by P@trick Haslum.

I would like to gratefully acknowledge the contributions of these researchers, and thank them for generously permitting me to use aspects of their presentation material.

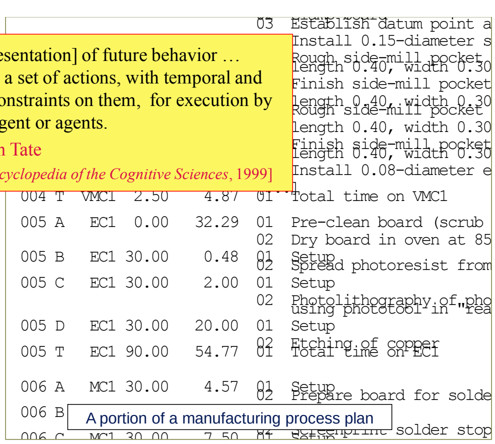




[a representation] of future behavior … usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents.

- Austin Tate

[*MIT Encyclopedia of the Cognitive Sciences*, 1999]



A portion of a manufacturing process plan

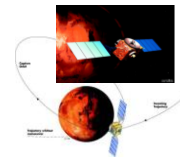
Modes of Planning

- Mixed Initiative Planning
- Automated Plan Generation

Example Planning Applications

Autonomous Agents for Space Exploration

- Autonomous planning, scheduling, control
  - NASA: JPL and Ames
- Remote Agent Experiment (RAX)
  - Deep Space 1
- Mars Exploration Rover (MER)



Other Autonomous Systems

Not necessarily embodied! →

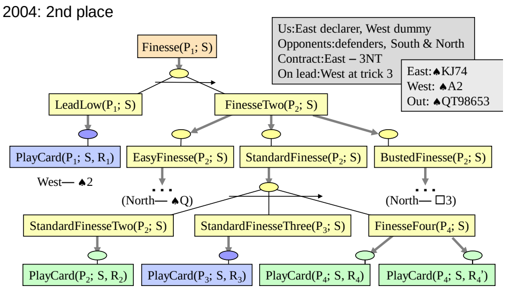
Manufacturing Automation

- Sheet-metal bending machines - Amada Corporation
  - Software to plan the sequence of bends
    [Gupta and Bourne, *J. Manufacturing Sci. and Engr.*, 1999]

Games

E.g., *Bridge Baron* - Great Game Products

- 1997 world champion of computer bridge
  [Smith, Nau, and Throop, *AI Magazine*, 1998]
- 2004: 2nd place



Other Applications

- Scheduling with Action Choices & Resource Requirements
  - Problems in supply chain management
  - HSTS (Hubble Space Telescope scheduler)
  - Workflow management
- Air Traffic Control
  - Route aircraft between runways and terminals. Crafts must be kept safely separated. Safe distance depends on craft and mode of transport. Minimize taxi and wait time.
- Character Animation
  - Generate step-by-step character behaviour from high-level spec
- Plan-based Interfaces
  - E.g. NLP to database interfaces
  - Plan recognition

Other Applications (cont.)

- Web Service Composition
  - Compose web services, and monitor their execution
  - Many of the web standards have a lot of connections to action representation languages
    - BPEL; BPEL-4WS allow workflow specifications
    - DAML-S allows process specifications
- Business Process Composition /Workflow Management
  - Including Grid Services/Scientific Workflow Management
- Genome Rearrangement
  - The relationship between different organisms can be measured by the number of “evolution events” (rearrangements) that separate their genomes
  - Find shortest (or most likely) sequence of rearrangements between a pair of genomes

Other Applications (cont.)

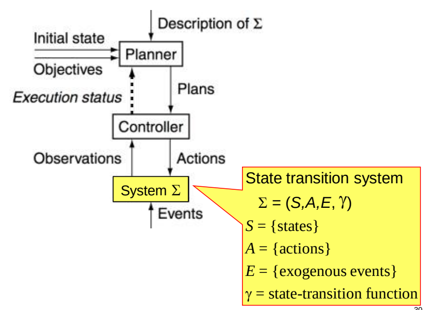
- Narrative generation
- Narrative understanding
- HCI/Dialogue planning
- Automated diagnosis
- …

Outline

- ➔ Conceptual model for planning
- Classes of planning problems
- Classes of planners and example instances
- Beyond planning
- Planning research – the big picture
- Some of what I hope you’ll get from the course

Conceptual Model

1. Environment



State Transition System

$\Sigma = (S,A,E,\gamma)$

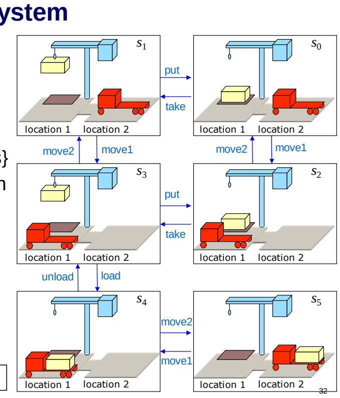
- S = {states}
- A = {actions}
- E = {exogenous events}
- State-transition function
  $\gamma : S \times (A \cup E) \rightarrow 2^S$
  - $S = \{s_0, \ldots, s_5\}$
  - A = {move1, move2, put, take, load, unload}
  - $E = \{\}$
  - γ: see the arrows

State Transition System

$\Sigma = (S,A,E,\gamma)$

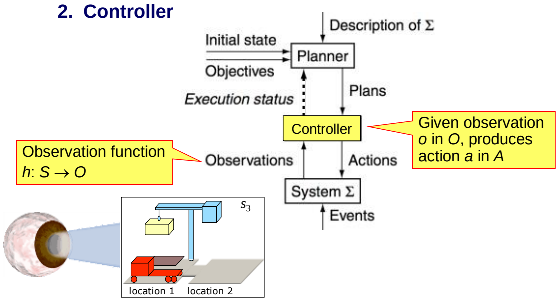
- S = {states}
- A = {actions}
- E = {exogenous events}
- State-transition function
  $\gamma : S \times (A \cup E) \rightarrow 2^S$
  - $S = \{s_0, \ldots, s_5\}$
  - A = {move1, move2, put, take, load, unload}
  - $E = \{\}$
  - γ: see the arrows

Dock Worker Robots (DWR):



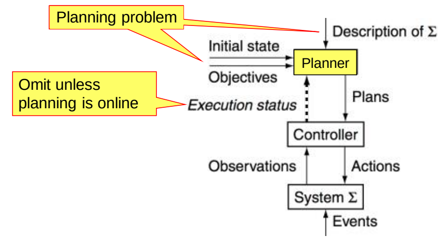
Conceptual Model

2. Controller



Conceptual Model

3. Planner's Input



Planning Problem

$P = (\Sigma, s_0, G)$

Σ: System Description

s_0: Initial state(s)
E.g., Initial state = s_0

G: Objective
Goal state,
Set of goal states,
Set of tasks,
"trajectory" of states,
Objective function, …
E.g., Goal state = s_5

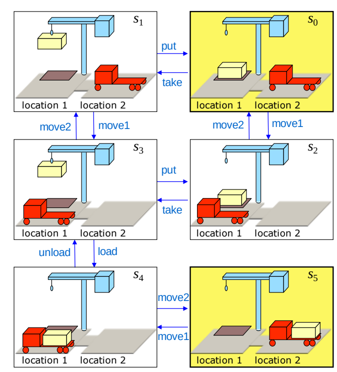
Planning Problem

$P = (\Sigma, s_0, G)$

Σ: System Description

s_0: Initial state(s)
E.g., Initial state = s_0

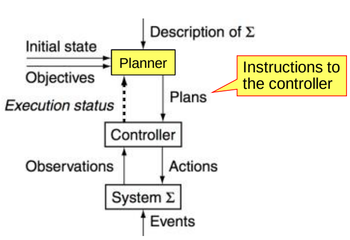
G: Objective
Goal state,
Set of goal states,
Set of tasks,
"trajectory" of states,
Objective function, …
E.g., Goal state = s_5



The Dock Worker Robots (DWR) domain

Conceptual Model

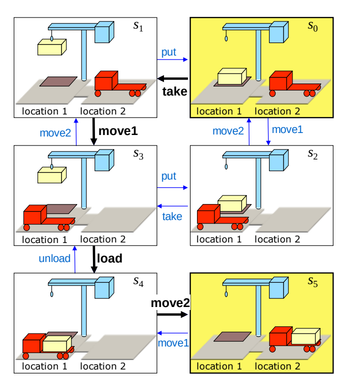
4. Planner's Output



Plans

Classical plan:
a sequence of actions
E.g.,
⟨take, move1, load, move2⟩

Policy:
partial function from S into A
E.g.,
$\{(s_0, \text{take}),$
$(s_1, \text{move1}),$
$(s_3, \text{load}),$
$(s_4, \text{move2})\}$



The Dock Worker Robots (DWR) domain

Outline

- Conceptual model for planning
- ➔ Classes of planning problems
- Classes of planners and example instances
- Beyond planning
- Planning research – the big picture
- Some of what I hope you'll get from the course

Planning

Agent: single agent or multi-agent
State:
complete or ncomplete (logical/probabilistic)
state of the world and/or agent's state of knowledge
Actions:
world-altering and/or knowledge-altering (e.g. sensing)
deterministic or non-deterministic (logical/stochastic)
Goal Condition:
satisficing or optimizing
final-state or temporally extended/control knowledge/script
optimizing: preferences or cost or utility or …
Reasoning:
offline or online (fully observable, partially observable)
Plans:
sequential, partial order, conformant, contingent, conditional (controller)

Different Classes Planning Problems

Varying components of the planning problem specification yields different classes of problems. E.g.,

dynamics: deterministic, nondeterministic, probabilistic
observability: full, partial, none
horizon: finite, infinite
objective requirement: satisfying, optimizing
…

Different Classes Planning Problems

dynamics: **deterministic**, nondeterministic, probabilistic
observability: full, partial, **none**
horizon: **finite**, infinite
objective requirement: **satisfying**, optimizing
…

- **classical planning**
- conditional planning with full observability (FOND)
- conditional planning with partial observability (POND)
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, **nondeterministic**, probabilistic
observability: **full**, partial, none
horizon: **finite**, **infinite**
objective requirement: **satisfying**, optimizing
…

- classical planning
- **conditional planning with full observability (FOND)**
- conditional planning with partial observability (POND)
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, **nondeterministic**, probabilistic
observability: full, **partial**, none
horizon: **finite**, **infinite**
objective requirement: **satisfying**, optimizing
…

- classical planning
- conditional planning with full observability
- **conditional planning with partial observability**
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, **nondeterministic**, probabilistic
observability: full, partial, **none**
horizon: **finite**, infinite
objective requirement: **satisfying**, optimizing
…

- classical planning
- conditional planning with full observability
- conditional planning with partial observability
- **conformant planning**
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, nondeterministic, **probabilistic**
observability: **full**, partial, none
horizon: **finite**, **infinite**
objective requirement: satisfying, **optimizing**
…

- classical planning
- conditional planning with full observability
- conditional planning with partial observability
- conformant planning
- **markov decision processes (MDP)**
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: deterministic, nondeterministic, **probabilistic**
observability: full, **partial**, none
horizon: **finite**, **infinite**
objective requirement: satisfying, **optimizing**
…

- classical planning
- conditional planning with full observability
- conditional planning with partial observability
- conformant planning
- markov decision processes (MDP)
- **partial observable MDP (POMDP)**
- preference-based/over-subscription planning

Different Classes Planning Problems

dynamics: **deterministic,** nondeterministic, probabilistic
observability: full, partial, **none**
horizon: **finite**, **infinite**
objective requirement: satisfying, **optimizing**
…

- classical planning
- conditional planning with full observability
- conditional planning with partial observability
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- **preference-based/over-subscription planning**

Other Dimensions

dynamics: deterministic, nondeterministic, probabilistic
: explicit time, implicit time
: instantaneous, durative
: continuous, discrete, hybrid
agents: multi-agent
perception: perfect, noisy
horizon: finite, infinite
objective requirement: satisfying, optimizing
objective form: final-state goal, temporally-extended goal, control knowledge, hierarchical task network (HTN), script/program (Golog)
plan form: sequential plan, partial order plan, controller, generalized plan, program…
…

- classical planning
- conditional planning with full observability
- conditional planning with partial observability
- conformant planning
- markov decision processes (MDP)
- partial observable MDP (POMDP)
- preference-based/over-subscription planning

Why is Planning Difficult?

- solutions to classical planning problems are paths from an initial state to a goal state in the transition graph
  - Efficiently solvable by Dijkstra's algorithm in $O(|V| \log |V| + |E|)$ time
  - Why don't we solve all planning problems this way?
- state space may be huge: 10^9, 10^{12}, 10^{15}, …states
- constructing the transition graph is infeasible!
- planning algorithms try to avoid constructing whole graph
- planning algorithms often are – but not guaranteed to be more effiencient that obvious solution methods constructing the transition graph and using e.g., Dijkstra's algorithm

Outline

- Conceptual model for planning
- Classes of planning problems
- ➔ Classes of planners and example instances
- Beyond planning
- Planning research – the big picture
- Some of what I hope you'll get from the course

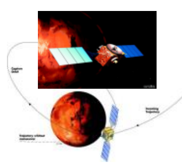
Three Main Classes of Planners

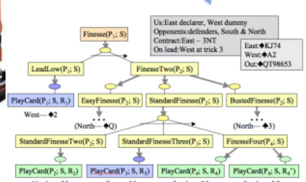
1. Domain-specific
2. Domain-independent
3. Domain-customizable

** Ghallab, Nau, and Traverso's use "configurable" (which I don't like)*
Also called "Domain-specific" or "Knowledge-Based"

1. Domain-Specific Planners

- Made or tuned for specific domain
- Won't work well (if at all) in any other domain
- Many successful real-world planning systems work this way





2. Domain-Independent Planners

- In principle, a domain-independent planner works in any planning domain
- Uses no domain-specific knowledge except the definitions of the basic actions

2. Domain-Independent Planners

- In practice,
  - Not feasible to develop domain-independent planners that work in *every* possible domain
- Make simplifying assumptions to restrict the set of domains
  - *Classical planning*
  - Historical focus of most automated-planning research

Very active area of research. Many excellent planning systems.

Restrictive Assumptions

- **A0: Finite system:**
  - finitely many states, actions, events
- **A1: Fully observable:**
  - the controller always knows the system's current state
- **A2: Deterministic:**
  - each action has only one outcome
- **A3: Static** (no exogenous events):
  - changes only occur as the result of the controller's actions
- **A4: Attainment goals:**
  - a set of goal states S_g
- **A5: Sequential plans:**
  - a plan is a linearly ordered sequence of actions ($a_1, a_2, \ldots a_n$)
- **A6: Implicit time:**
  - Actions are instantaneous (have no duration)
- **A7: Off-line planning:**
  - planner doesn't know the execution status

Classical Planning

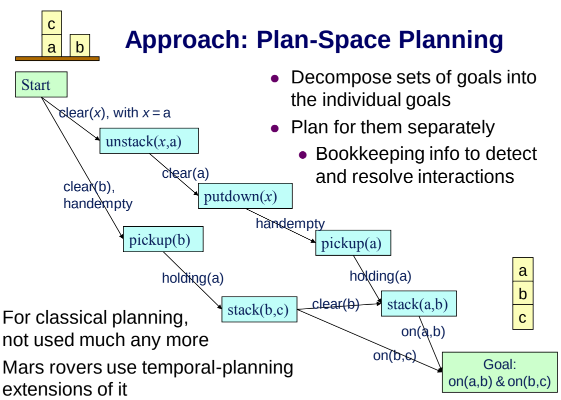
- Classical planning requires all eight restrictive assumptions
  - Offline generation of action sequences for a deterministic, static, finite system, with complete knowledge, attainment goals, and implicit time
- Reduces to the following problem:
  - Given (Σ, s_0, G)
  - Find a sequence of actions (a_1, a_2, … a_n) that produces a sequence of state transitions (s_1, s_2, …, s_n) such that $G$is in s_n.
- This is just path-searching in a graph
  - Nodes = states
  - Edges = actions
- ***Is this trivial?***

Classical Planning (cont.)

It's hard because problems are huge!

- Generalize the earlier example:
  - 5 locations, 3 robot carts, 100 containers, 3 piles
    - Then there are 10^{277} states
- Number of particles in the universe is only about 10^{87}
  - The example is more than 10^{190} times as larger!
- Automated planning research has been heavily dominated by domain-independent classical planning
  - Dozens of different algorithms
  - We'll cover the state-of-the-art in this area

Approach: Plan-Space Planning



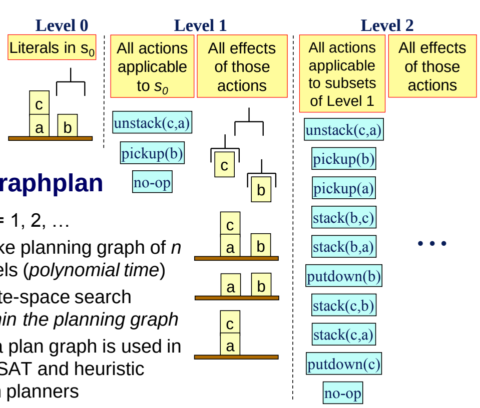
- Decompose sets of goals into the individual goals
- Plan for them separately
  - Bookkeeping info to detect and resolve interactions

For classical planning, not used much any more

Mars rovers use temporal-planning extensions of it

Approach: Planning Graphs

- Relaxed problem [Blum & Furst, 1995]
- Apply all applicable actions at once
- Next "level" contains all the effects of all of those actions



E.g., Graphplan

- For n = 1, 2, …
  - Make planning graph of n levels (*polynomial time*)
  - State-space search *within the planning graph*
- Such a plan graph is used in many SAT and heuristic search planners

Approach: Heuristic Search

- Can we do an A*-style heuristic search?
- Historically, it was difficult to find a good h function
  - Planning graphs make it feasible
    - Can extract h from the planning graph
- Problem: A* quickly runs out of memory
  - So do a greedy search
- Greedy search can get trapped in local minima
  - Greedy search plus local search at local minima
- HSP [Bonet & Geffner], FastForward (FF) [Hoffmann], Fast Downward [Helmert], LAMA [Richter], etc.

Approach: Translation to General Problem Solver

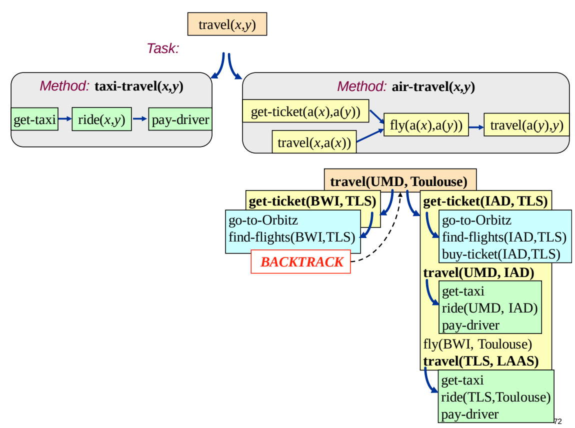
- Translate the planning problem or the planning graph into another kind of problem for which there are efficient solvers
  - Find a solution to that problem
  - Extract the plan from the solution
- SAT solvers
  - SATplan and Blackbox [Kautz & Selman]
- Answer Set Programming (ASP) solvers
  - [Son *et al.*], [Lifschitz *et al.*], etc.
- Integer programming solvers such as Cplex
  - [Vossen *et al.*]

3. Domain-customizable

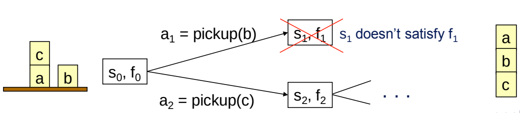
- Domain-independent planners are quite slow compared with domain-specific planners
  - Blocks world in linear time [Slaney and Thiébaux, *A.I.*, 2001]
  - Can get analogous results in many other domains
- But don't want to write a new planner for every domain!
- **Domain-customizable planners**
  - Domain-independent planning engine
  - Input (the "objective") includes info about how to solve problems in the domain.
    - Hierarchical Task Network (HTN) planning
    - Planning with control formulas
    - Planning with a plan script or agent program

Approach: HTN Planning

- Problem reduction
  - *Tasks* (activities) rather than goals
  - *Methods* to decompose tasks into subtasks
  - Enforce constraints, backtrack if necessary
- Real-world applications
- Noah, Nonlin, O-Plan, SIPE, SIPE-2,SHOP, SHOP2



Approach: Planning with Control Formulas



- At each state s_i we have a *control formula* f_i in temporal logic

  $$ontable(x) \wedge \neg\exists[y{:}\mathrm{GOAL}(on(x,y))] \Rightarrow \bigcirc(\neg holding(x))$$

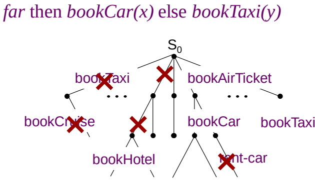
  "never pick up *x* from table unless *x* needs to be on another block"
- For each successor of *s*, derive a control formula using *logical progression*
- Prune any successor state in which the progressed formula is false
  - TLPlan [Bacchus & Kabanza]
  - TALplanner [Kvarnstrom & Doherty]

Approach: Planning w/ Program or Plan Script

E.g., **Golog** [Levesque et al.]
Nondeterministic programs that act as procedural control knowledge, placing constraints on the valid action sequence/plans

E.g., *bookAirTicket(x);* if *far* then *bookCar(x)* else *bookTaxi(y)*

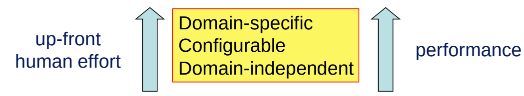
procedural constructs:
- sequence
- if-then-else
- nondeterministic choice
- while-do, etc.



Three Main Classes of Planners

1. Domain-specific
2. Domain-independent
   E.g.,
   Planning graph-based, SAT-based, heuristic search
3. Domain-customizable
   E.g.,
   HTN, domain control formula, agent programs/scripts

Comparisons (in general)



- Domain-specific planner
  - Write an entire computer program - lots of work
  - Lots of domain-specific performance improvements
- Domain-independent planner
  - Just give it the basic actions - not much effort
  - Can be less efficient (but not always)!

Outline

- Conceptual model for planning
- Classes of planning problems
- Classes of planners and example instances
- ➔ Beyond planning
- Planning research – the big picture
- Some of what I hope you'll get from the course

Broad Application of Planning Techniques

Planning algorithms are applicable to a broad range of applications that can roughly be viewed as reachability problems. E.g.,

- Software verification
- Diagnosis of dynamical systems
- Story understanding
- Situation assessment/Plan recognition
- Gene rearrangement

Outline

- Conceptual model for planning
- Classes of planning problems
- Classes of planners and example instances
- Beyond planning
- ➔ Planning research
- Some of what I hope you'll get from the course

Planning Research – The big picture

Two research communities that make fundamental contributions to research in planning:

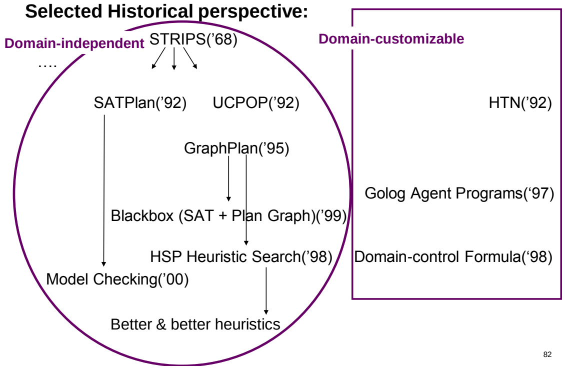
1) Knowledge Representation and Reasoning Community
   - mathematical foundations of planning
   - knowledge representation, formal properties, etc.

2) Automated Planning and Scheduling Community
   - Driven largely by the objective of developing fast and effective planning algorithms

We will look at research from both communities.

Planning Algorithms – Recent Advances

Selected Historical perspective:



Planning Research – The big picture

The Landscape:

CONFERENCES

ICAPS* (Int. Conf on AI Planning and Scheduling)
*merging of AIPS and ECP
AAMAS (Int. Conf. on Autonomous Agents and Multiagent Systems)
KR
IJCAI, AAAI, ECAI

JOURNALS

JAIR, AIJ

BIENNIAL COMPETITION and BENCHMARKING DOMAINS

IPC-*n* (International Planning Competition)
PDDL (Planning Domain Definition Language)
standard input language for most benchmark problem sets

Planning Research – The big picture

Recent Advances

Very "active" field -- lots of papers in top conferences

- Tremendous strides in deterministic plan synthesis
  - Biennial Intl. Planning Competitions
- Current interest is in exploiting the insights from deterministic planning techniques to other planning scenarios

Some topics of recent focus:

- Better heuristics
- Better search, real-time search, sampling, ….
- Richer domain customization (including preferences)
- From discrete to timed hybrid and/or continuous systems
- Planning and learning

Outline

- Conceptual model for planning
- Classes of planning problems
- Classes of planners and example instances
- Beyond planning
- Planning research – the big picture
- ➔ Some of what I hope you'll get from the course

What will you get from this course?

- big picture of different kinds of planning problems
- formal foundations of planning & reasoning about action that will help you study and innovate in this area.
- algorithms for solving different problem classes, with an emphasis on the classical ("simplest") setting:
  - algorithms based on heuristic search
  - algorithms based on SAT
  - algorithms that exploit rich objectives (domain control knowledge, temporally extended goals, preferences)
- many of these techniques are applicable to problems outside AI as well
- hands-on experience with a classical planner (optional)

For Thursday (May 15)

Read Chapter 1 of Geffner&Bonet book (available online)

***Start* skimming/reviewing Chapters 1, 2, and 4, 5 in the reference textbook.**

(The URL will be posted on our course web page.)

Important Announcement

- ✓ **Please add your name to the list that is circulating**
- ✓ **From time to time we will need to have a tutorial (especially for the assignment). There will be a doodle poll regarding scheduling of the tutorial. I'll schedule the tutorial hour at a time when all registered students can attend.**