Practical Planning

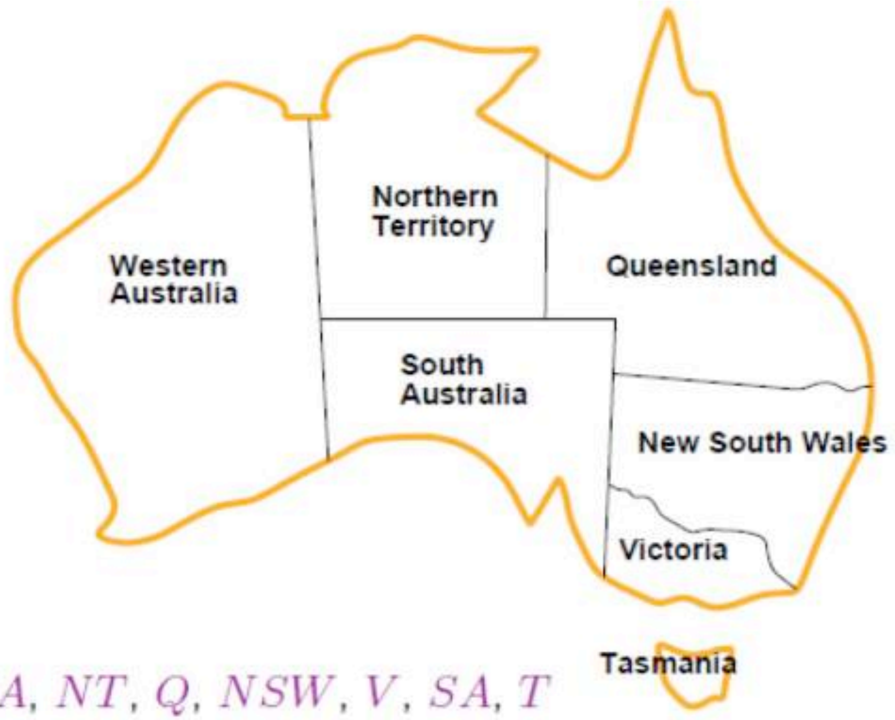
Minh Do
ERA/PARC

Automated Planning Approaches

- Domain-Independent Planning:
  - Systematic search: progression, regression, plan-space ← CS221
  - Graphplan: ← CS221
  - Local search
  - Compilation approach
- Domain-Specific Planning:
  - HTN Planning ← Lecture 16
  - Temporal Logic-based Planning

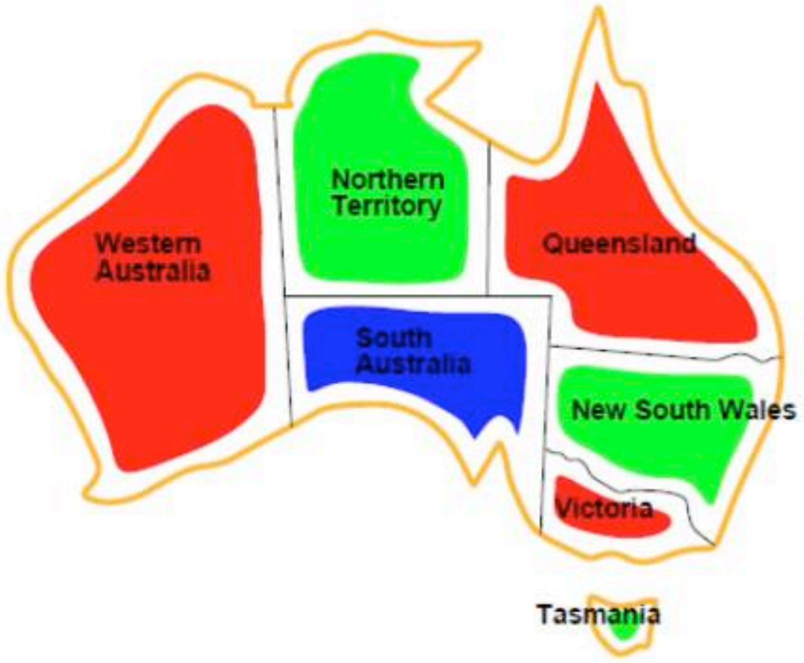
parc
Palo Alto Research Center

Constraint Satisfaction Problem: Map-Coloring [Lecture 14]



Variables WA, NT, Q, NSW, V, SA, T
Domains $D_i = \{red, green, blue\}$
Constraints: adjacent regions must have different colors
  e.g., $WA \neq NT$ (if the language allows this), or
  $(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \ldots\}$

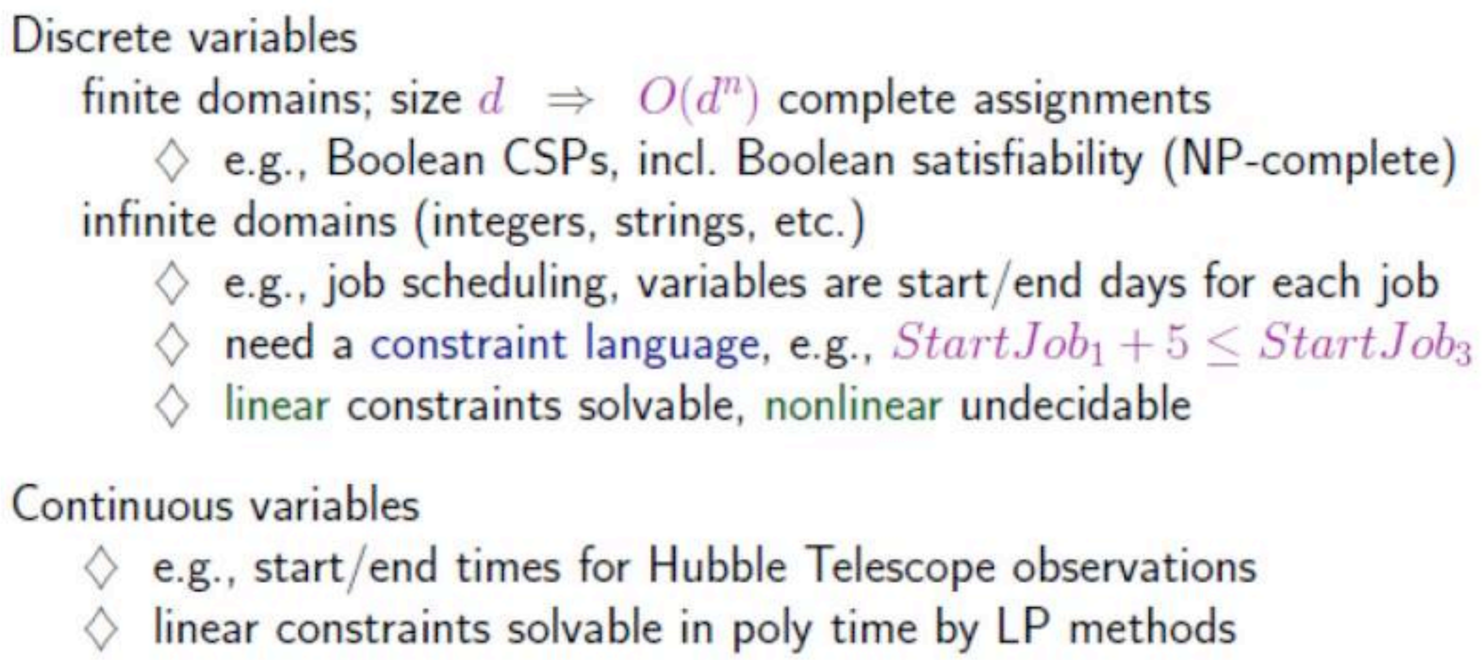
Constraint Satisfaction Problem: Map-Coloring [Lecture 14]



Solutions are assignments satisfying all constraints, e.g.,
$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

parc Palo Alto Research Center

Varieties of CSPs [Lecture 14]



Discrete variables

- finite domains; size $d \Rightarrow O(d^n)$ complete assignments
  - ◇ e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)
- infinite domains (integers, strings, etc.)
  - ◇ e.g., job scheduling, variables are start/end days for each job
  - ◇ need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
  - ◇ linear constraints solvable, nonlinear undecidable

Continuous variables

- ◇ e.g., start/end times for Hubble Telescope observations
- ◇ linear constraints solvable in poly time by LP methods

Common Problem Structure Setup:

What are the variables?
What are the domain of each variable?
What are the constraints between variables?

Outline

- Compilation Approaches for Planning
  - Satisfiability (SAT) (binary CSP – Lecture 14)
  - Constraint Satisfaction Problem (CSP)
  - Integer Linear Programming (ILP)
    (infinite domain, continuous variable – Lecture 14)
  - Answer Set Programming (ASP) (Lecture 11)

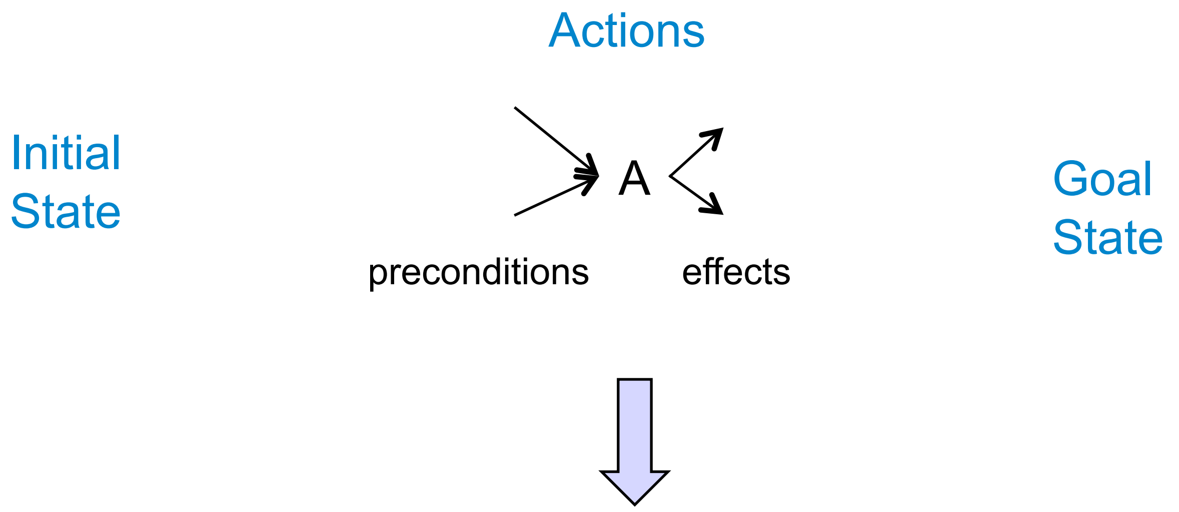
  → Define Variables/Domain/Constraints

- Planning Applications

parc
Palo Alto Research Center

Compiling Planning Problem to a Combinatorial Substrate

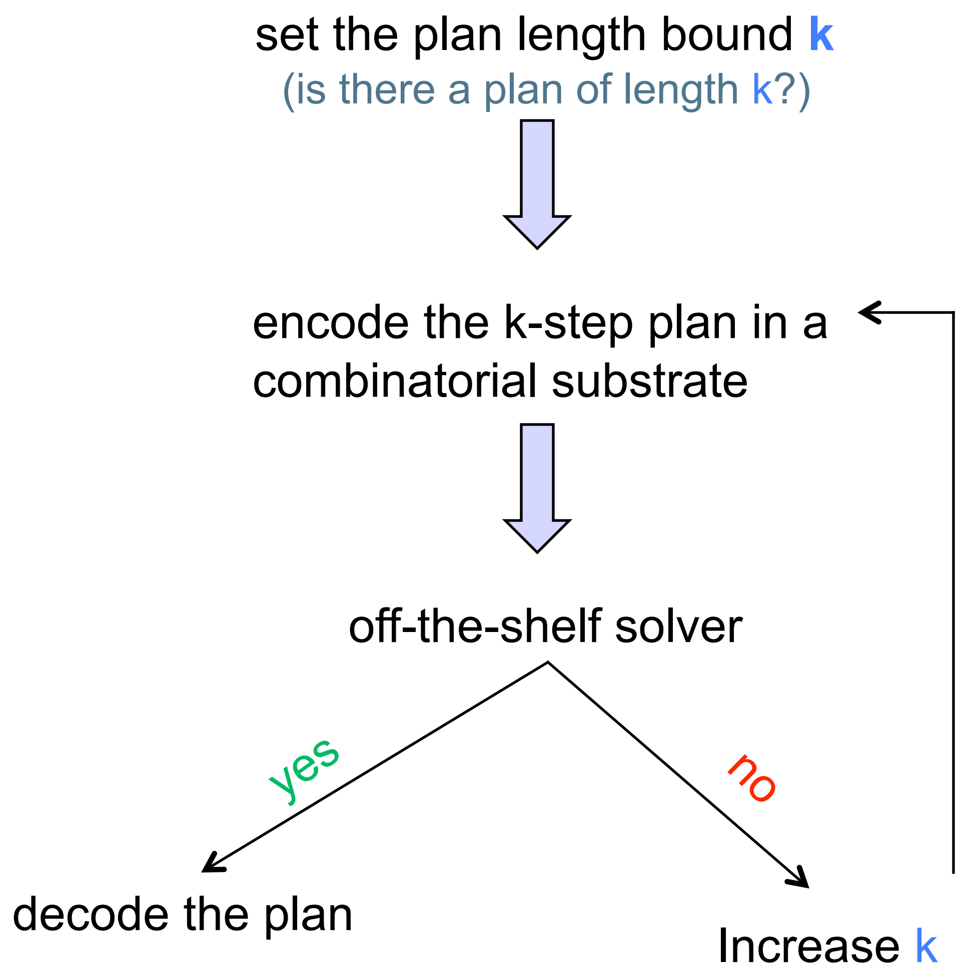
Planning Problem:



Compilation in CSP/SAT/ASP:

What are the variables?
What are the domain of each variable?
What are the constraints between variables?

Compilation Approach: Overview


set the plan length bound k
(is there a plan of length k?)
encode the k-step plan in a combinatorial substrate
off-the-shelf solver
yes
no
decode the plan
Increase k

Motivation:

X times improvements in state-of-the-art solver
(SAT/CSP competition)

$\Downarrow$ automatic

X times improvements in planner performance

parc
Palo Alto Research Center

Planning as Satisfiability (SAT)

- Most popular compilation/translation based approach
- SAT-based planners regularly compete in IPCs (despite the domination of search-based planners)
- A paper on new SAT-based planning technique won AAAI-2010 Best Paper Award

parc
Palo Alto Research Center

Satisfiability Problem

- Variables: True/False
- Constraints: AND, OR, NOT

  a /\ b

  a V b

  ~a
- Problem Representation:

  ((a v ((~b /\ c) v ~d) /\ e) …….) …..

  –Conjunctive Normal Form (CNF):

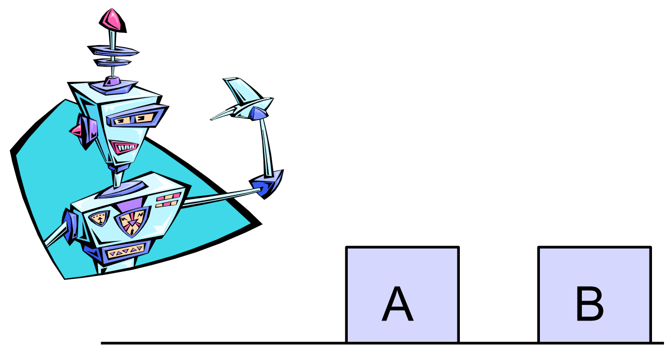
  » ~a /\ (b V c) /\ (d V ~e)

  –Disjunctive Normal Form (DNF):

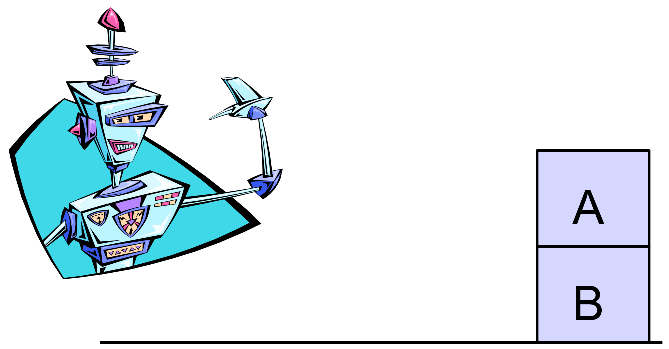
  » ~a V (b /\c) V (d /\ ~e)
- Problem: find a complete T/F variable assignment satisfying all constraints

Example

Initial State



Goal State



Fluents:

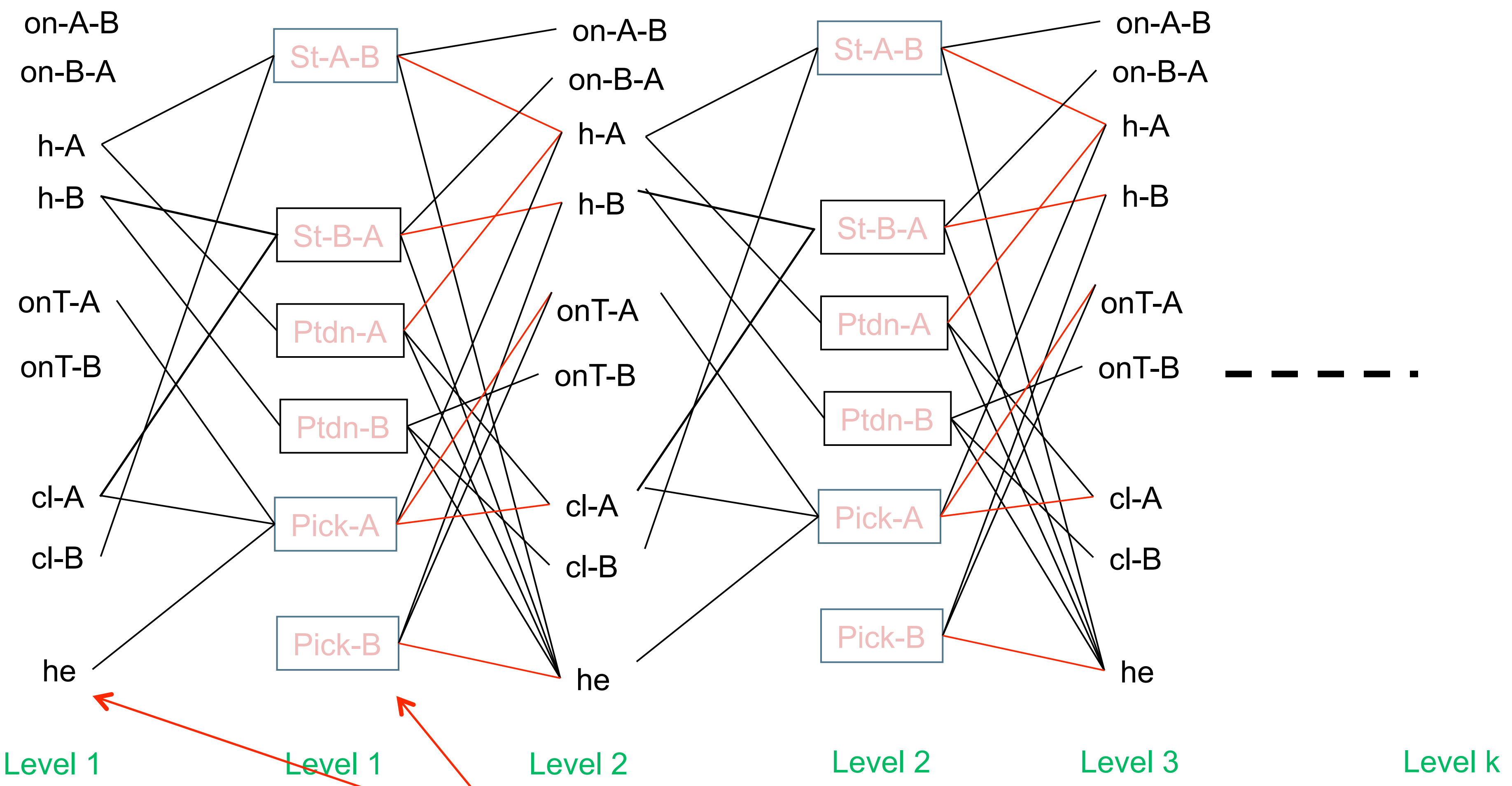
OnTable(A), OnTable(B)
On(A,B), On(B,A)
Clear(A), Clear(B)
Holding(A), Holding(B)
HandEmpy

Actions:

Pickup(A), Pickup(B)
Stack(A,B), Stack(B,A)
Putdown(A), Putdown(B)

parc Palo Alto Research Center

Naïve SAT Encoding



Use fluents describe what's true at each "level"

Variables

- l_i to denote the fluent of literal l in level i

  $l_2 = On(A,B,2)$

- a_i to denote if action a is the i^{th} step of the plan

  $a_1 = Pickup(A,1)$

Constraints

- **Formula describing the initial state:**

  $\bigwedge\{l_0 \mid l \in s_0\} \wedge \bigwedge\{\neg l_0 \mid l \in L - s_0\}$

- **Formula describing the goal state:**

  $\bigwedge\{l_n \mid l \in g^+\} \wedge \bigwedge\{\neg l_n \mid l \in g^-\}$

- **Formulas describing the preconditions and effects of actions:**

  For every action a in A, formulas describing what changes a would make if it were the i'th step of the plan:

  - $a_i \Rightarrow \bigwedge\{p_i \mid p \in \text{Precond}(a)\} \wedge \bigwedge\{e_{i+1} \mid e \in \text{Effects}(a)\}$

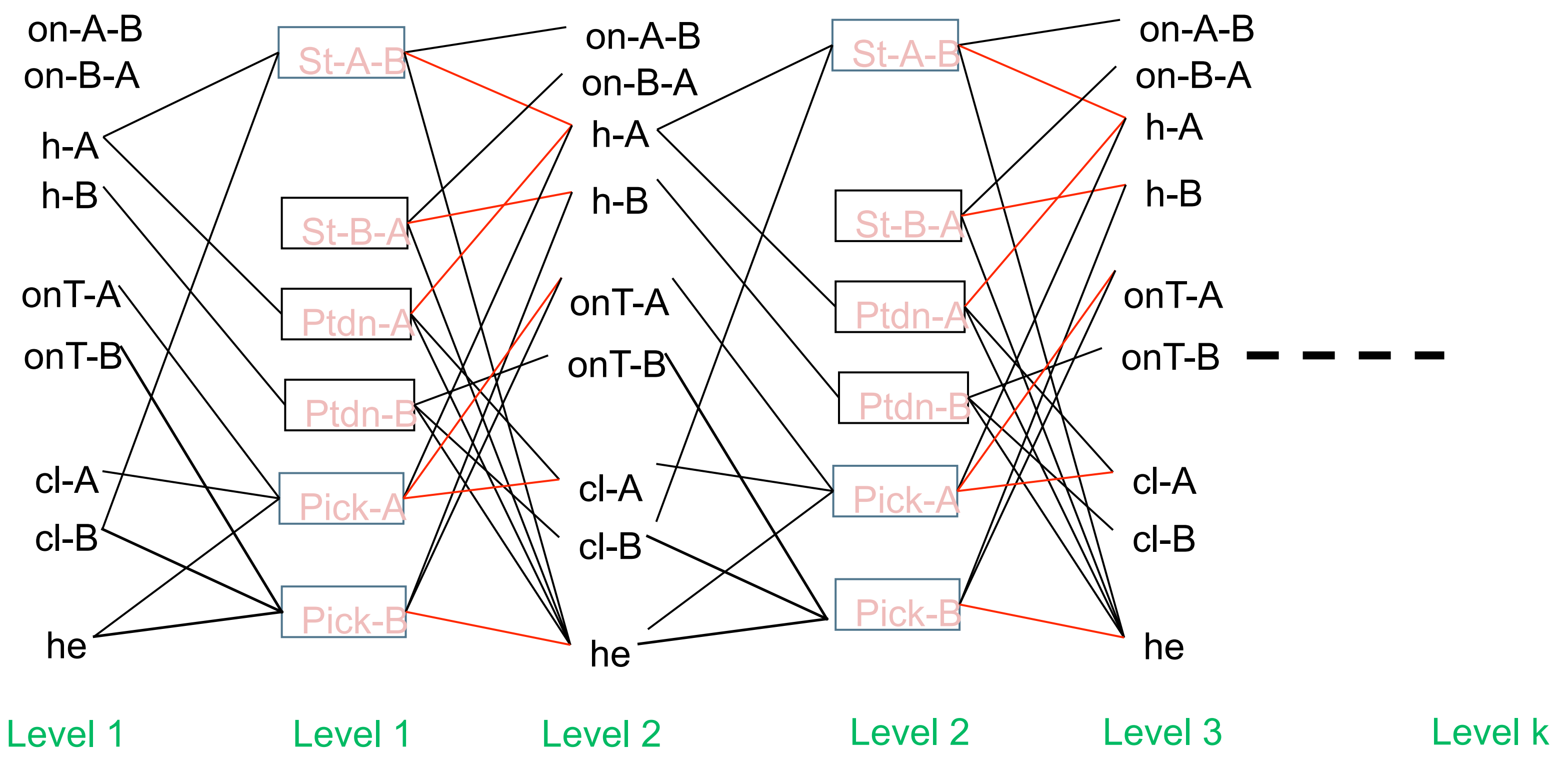
- **Formulas describing *Complete exclusion*:**
  - For all actions a and b, formulas saying they cannot occur at the same time

    $\neg a_i \vee \neg b_i$
  - this guarantees there can be only one action at a time

- **Formulas providing a solution to the Frame Problem**

  Explanatory frame axioms: $l_i \rightarrow l_{i-1} \vee (\bigvee a_{i-1}$: l is a's effect)

SAT Encoding: Problems



(1) Huge number of variables and constraints

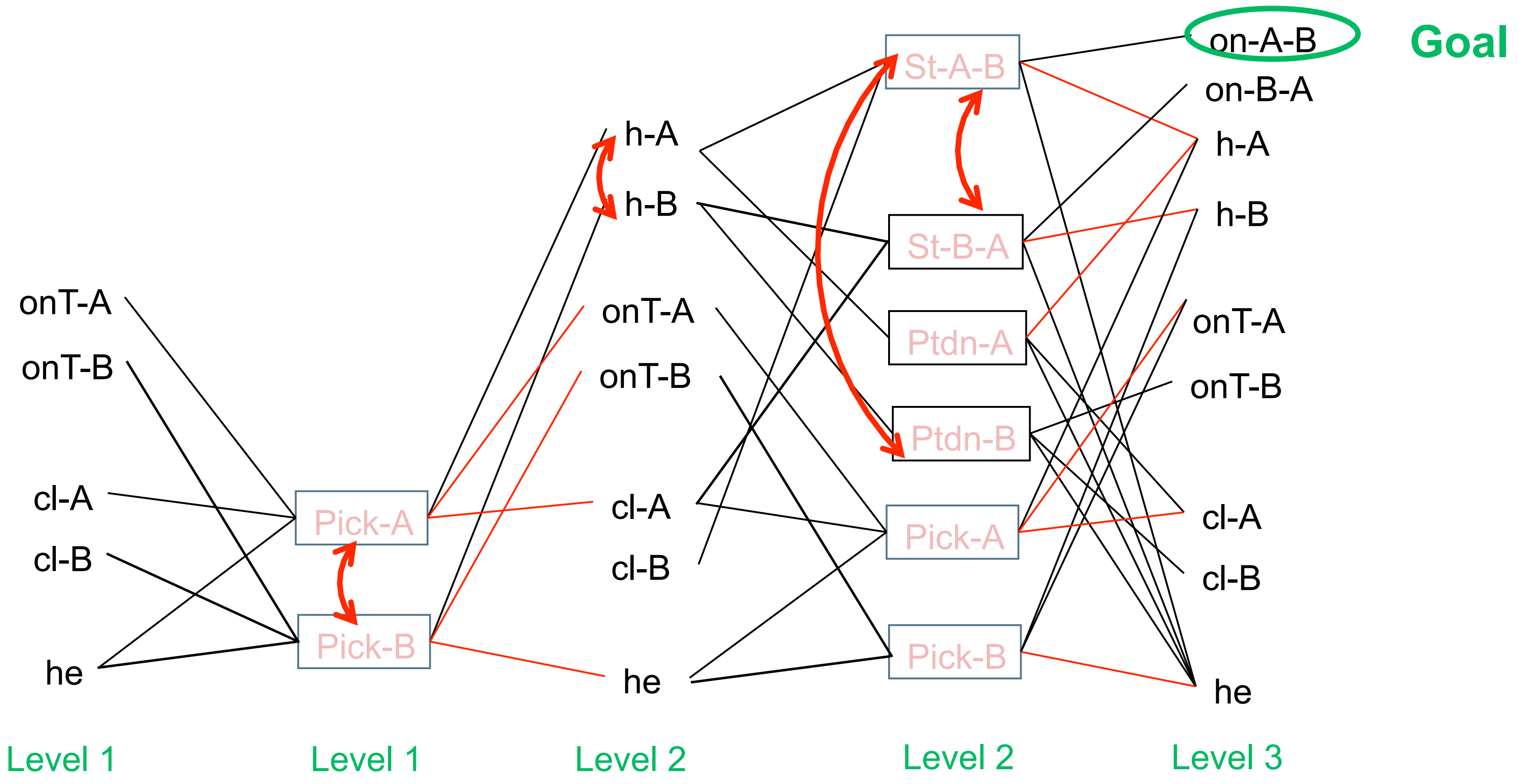
(2) Solve large number of SAT encodings (k = 1, 2, ….n)

Blackbox: Planning Graph + SAT

CS 221 – Lecture 9 – Slide #25 on Planning Graph

- Each level consists of
- ***Literals*** = all those that *could* be true at that time step, depending upon the actions executed at preceding time steps.
- ***Actions*** = all those actions that *could* have their preconditions satisfied at that time step, depending on which of the literals actually hold.

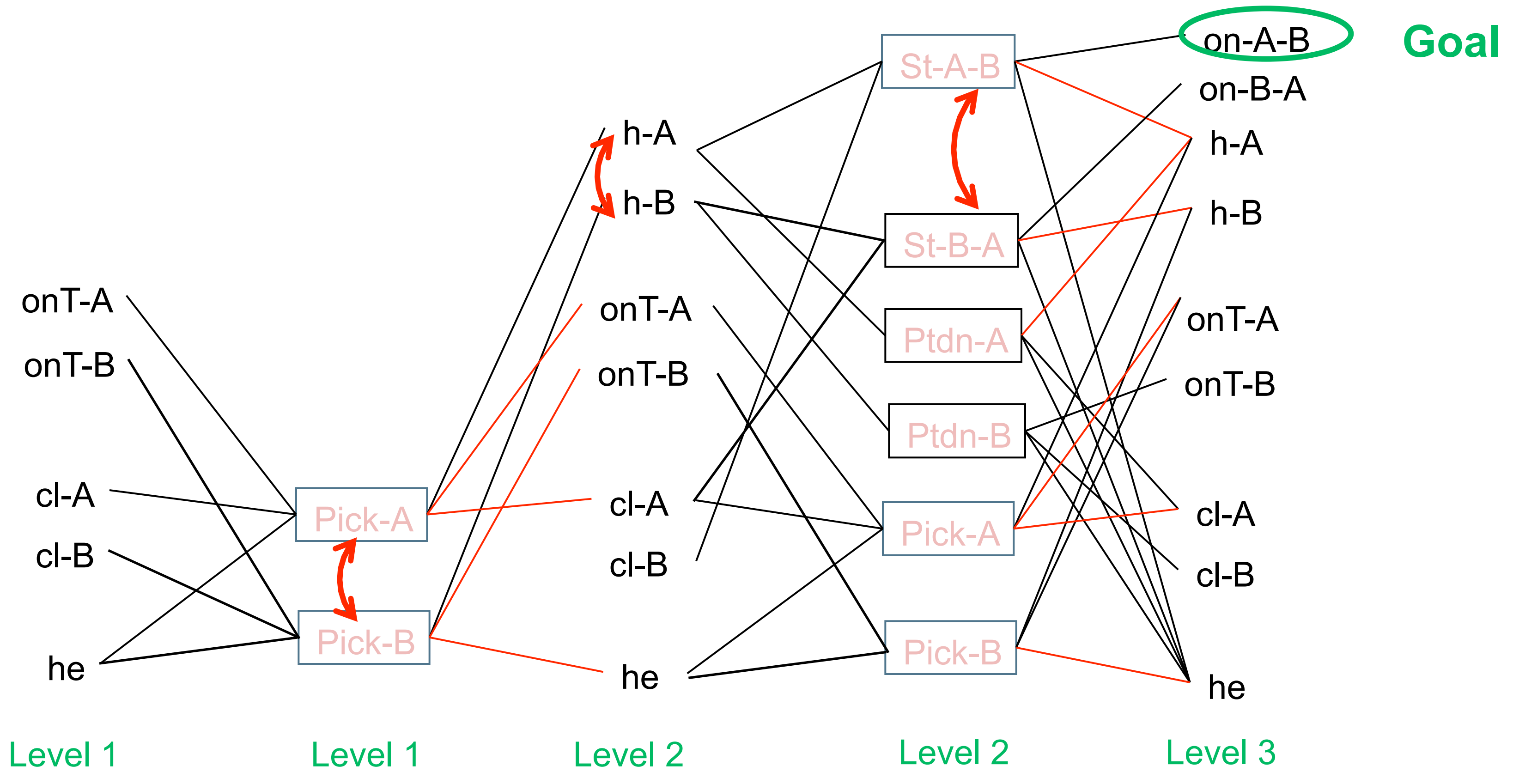
Planning Graph: Example



Mutex Propagation: facts and actions that cannot happen together

- Reduce the number of "reachable" actions & facts at each level
- Better estimation of planlength
- Mutex constraints help solver during constraint propagation

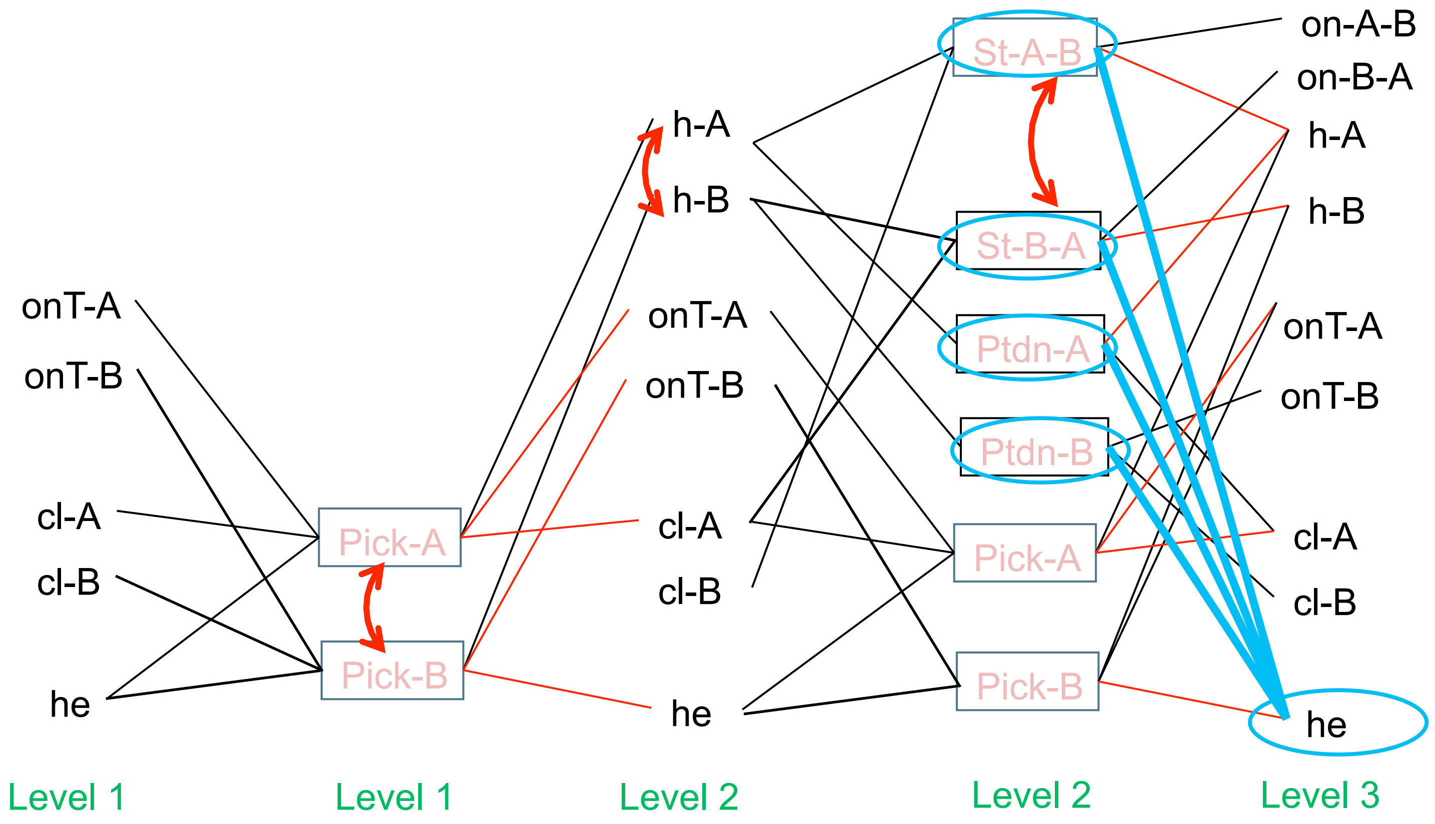
Blackbox: Planning Graph + SAT



Step 1: build the planning graph until all goals appear non-mutex
Step 2: backward relevant analysis to remove irrelevant actions/facts
Step 3: encode the remaining graph as SAT

(**Note**: There are other more recent techniques to improve SAT-based planners
(e.g., long-distance mutex, transition-based encoding)

GP-CSP: Planning Graph + (discrete) CSP



CSP Variable: State variable *s* with domain = actions in previous level supporting *s*

Constraints:

Activation: $he_3 = St\text{-}A\text{-}B_2 \rightarrow (h\text{-}A_2 \neq NULL) \wedge (cl\text{-}A_2 \neq NULL)$

Mutex: NOT $(he_3 = St\text{-}A\text{-}B_2 \wedge on\text{-}B\text{-}A_3 = St\text{-}B\text{-}A_2)$

Goals: $on\text{-}A\text{-}B_3 \neq NULL$

(discrete) CSP vs. SAT

- CSP encoding is generally (much) smaller
- CSP solver is more expressive → easier to extend the planner to more expressive planning problem
  - → But people have found creative ways to use SAT for planning with continuous resources and preferences
- More suitable for newer “multi-value” planning representation

- SAT solvers advance much faster
  - –SAT planner is generally faster (now)

parc Palo Alto Research Center

Integer Linear Programing

- SAT T/F variables → ILP 0/1 variables
- SAT constraints → ILP constraints

  a v b v c → $a + b + c \geq 1$

  a /\ b /\ c → $a + b + c = 3$

- Advantages:
  - –Some constraints are much more compactly represented in ILP:
    - » Only one action in a given level (XOR): $a_1 + a_2 + \ldots + a_n = 1$
  - –Can represent continuous variable naturally (e.g., robot battery level)
  - –Advance ILP encoding use bi-level graph with variables beyond 0/1 → even more compact representation

Answer Set Planning [Lecture 11]

- Planning problem $\langle A, I, G \rangle$
  - A – a set of action descriptions
  - I – initial state
  - G – goal state

$\rightarrow$ Logic program $P(A,I,G)$ – Three different sets of rules:

  - Representing A and I
  - Representing G
  - Generating action occurrences

- Each answer set of $P(A,I,G)$ corresponds to a trajectory achieving G and vice versa.

Answer Set Planning – Example

- *⟨A,I,G⟩*
  - Action theory: *drive* **causes** *at(airport)* **if** *at(home)*
    drive **executable_if** *hasCar*
  - Initially: *at(home), hasCar*
  - Goal: *at(airport)*
- Logic program *P(A,I,G)*

$$\left.\begin{array}{l} holds(at(airport), T+1) \leftarrow holds(at(home),T), holds(hasCar, T), occ(drive, T) \\ holds(F, T+1) \leftarrow holds(F, T), not\ holds(\neg F, T+1). \end{array}\right\} A$$

$1 \{ occ(A,T) : action(A) \} 1 \leftarrow time(T)$ ⟵ Action occurrences

$$\left.\begin{array}{l} holds(at(home), 0). \\ holds(hasCar, 0). \end{array}\right\} I$$

$\leftarrow not\ holds(at(airport), plan\_length).$ ⟵ Goal

parc Palo Alto Research Center

[slide from Son Tran@UNM]

Advantage of ASP Compilation

- Shown to be easy to extend to more complex planning problems:
  - Uncertainty: conformant, contingent planning (with sensing actions)
  - Planning with qualitative preferences

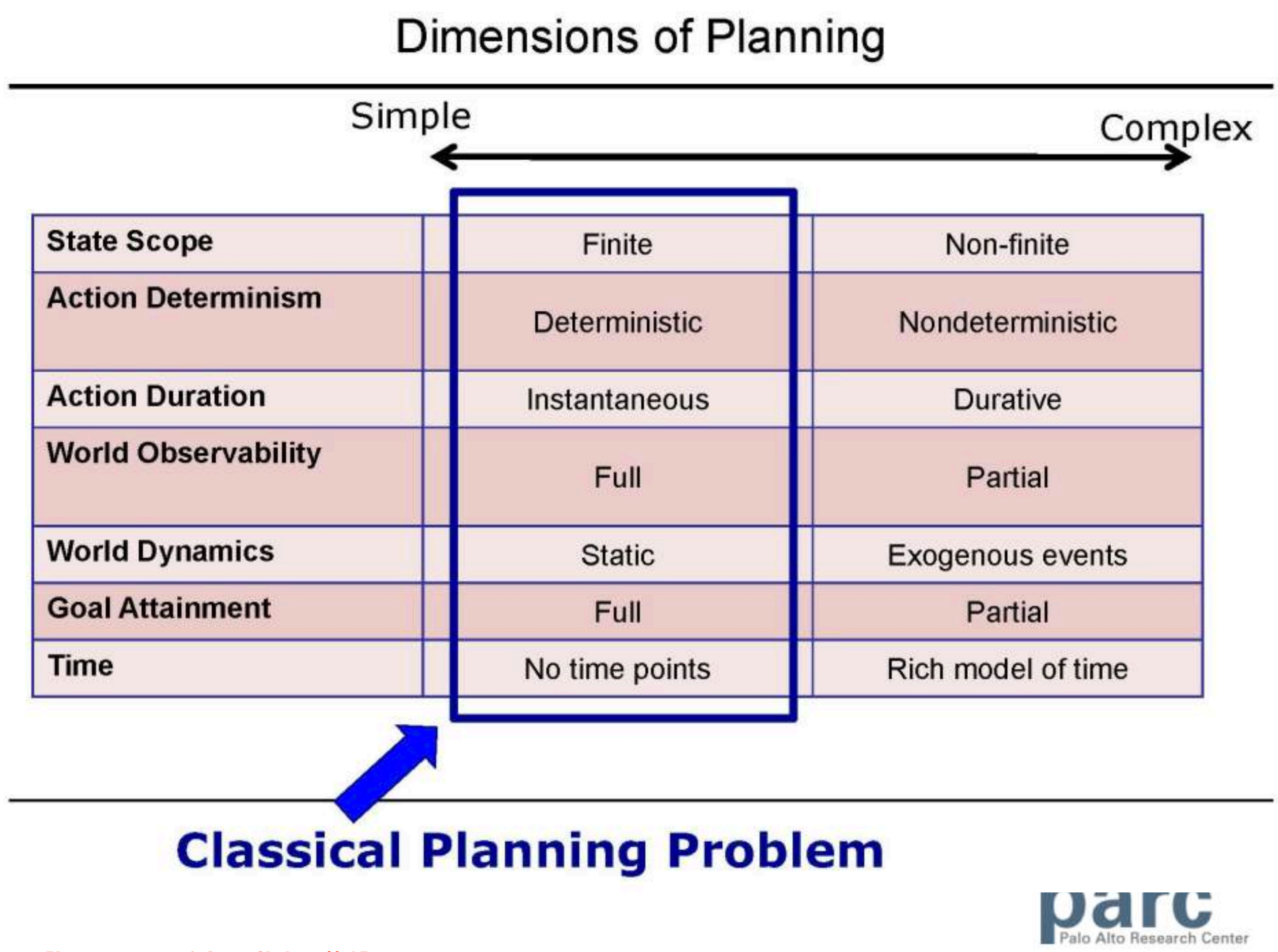
Outline

- Compilation approach for Planning
  - Satisfiability (SAT) (binary CSP – Lecture 14)
  - CSP
  - MILP (infinite domain, continuous variable – Lecture 14)
  - ASP
- Planning applications

ICAPS 2011 Stats

- 12/47 accepted papers are applications
- 13 papers accepted at Scheduling and Planning Applications Workshop (SPARK)
- 17/20 system demos are applications

parc
Palo Alto Research Center



Dimensions of Planning

Simple ⟷ Complex

State Scope	Finite	Non-finite
Action Determinism	Deterministic	Nondeterministic
Action Duration	Instantaneous	Durative
World Observability	Full	Partial
World Dynamics	Static	Exogenous events
Goal Attainment	Full	Partial
Time	No time points	Rich model of time

Classical Planning Problem

parc Palo Alto Research Center

[Lecture 16, slide #4]

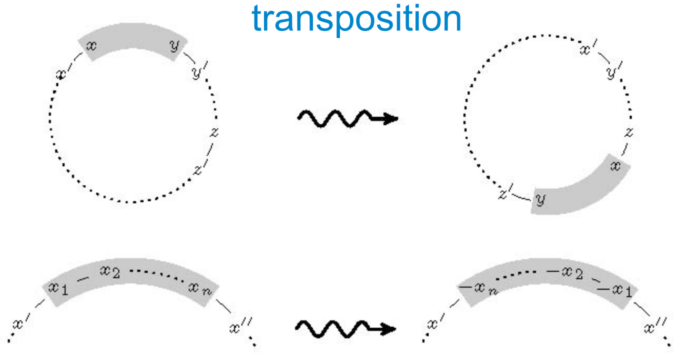
Classical Planning: Application

- Simplest form of planning:
  - More complex planning domains can be “relaxed” and solved by classical planner

- Applications:
  - Games: iceblock, sokoban, freecell
  - Diagnosis as planning
  - Greenhouse logistic
  - Genome rearrangement
  - Analyzing computer network vulnerability
  - Military training

Genome Rearrangement

[Erdem & Tillier, 2005; Haslum, 2010]



Genome edit operation = planning action

Problem: finding minimum edit distance = finding shortest plan
→ help build the most plausible "evolutionary tree"

Analyzing Computer Network Security

[Boddy et al., 2005]

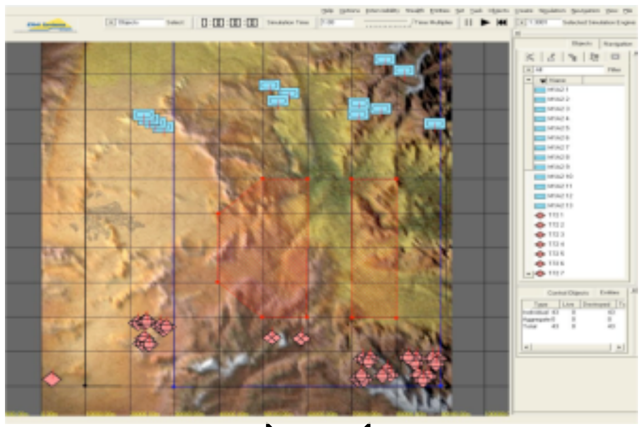
The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

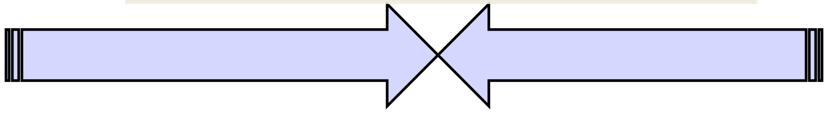
Plan = adversary course of action to exploit a given system vulnerability

•Find all plans (attack tree/graph)
•Find way to "fix/prevent" attack plans
(metric FF planner was used ← **Lecture 16**)

parc
Palo Alto Research Center

SHOGUN: Military Training





Military Trainee
(blue)

war gaming

SHOGUN (red)
(classical planner FD)

Domain Characteristics: temporal, non-deterministic action, partial observability

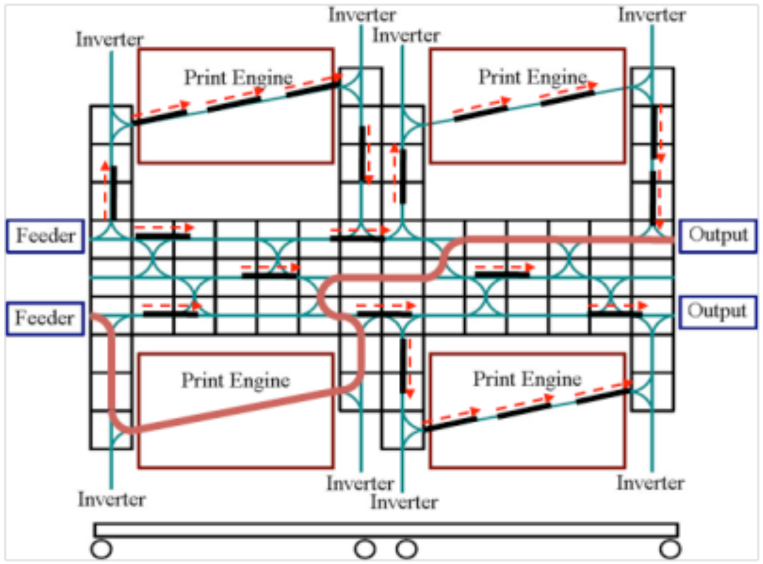
→ Clever "relaxation" scheme to map to use classical planner
-**Stochastic action effects**: ignore all possible effects except most likely one
-**Partial observability**: "optimistic sensing" assume no blue force found for all red sensing actions

Application: Temporal Planning/ Scheduling

- Multi-modal logistic
- Mars rovers
- Satellite coordination
- Robotic task planning
- Applications at ERA/PARC:
  - Multi-engine printer
  - LCD manufacturing plan
  - Automated warehouse

PARC: Multi-Engine Printer

220 pages/minute

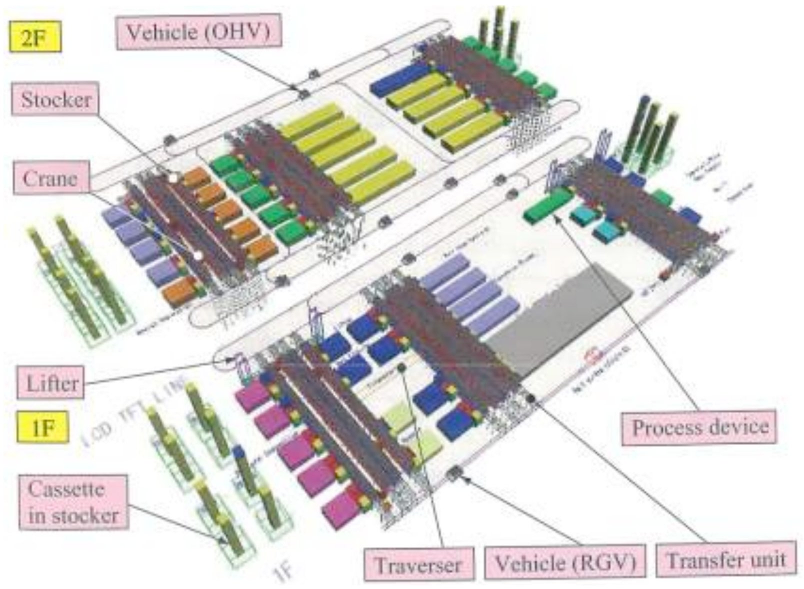


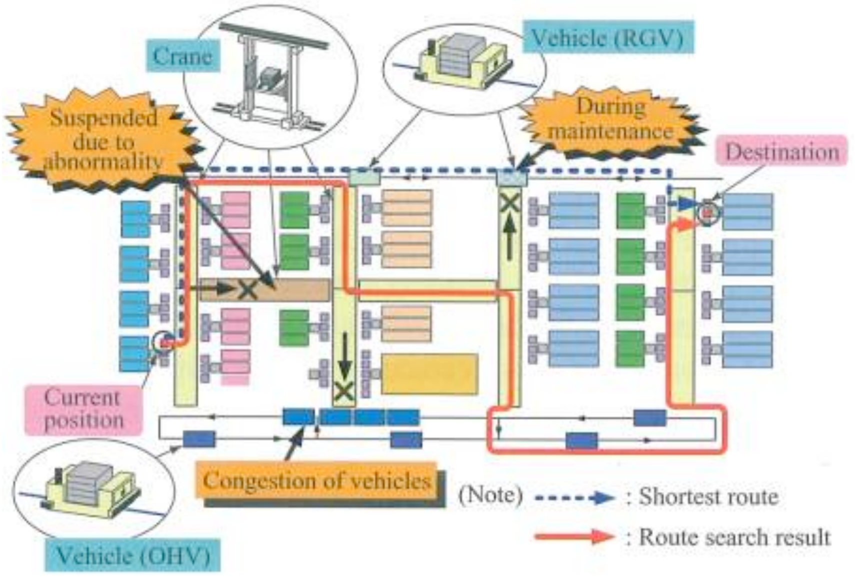
180 pages/minute

Find paper routes that run arbitrary printer configurations at maximum productivity

parc Palo Alto Research Center

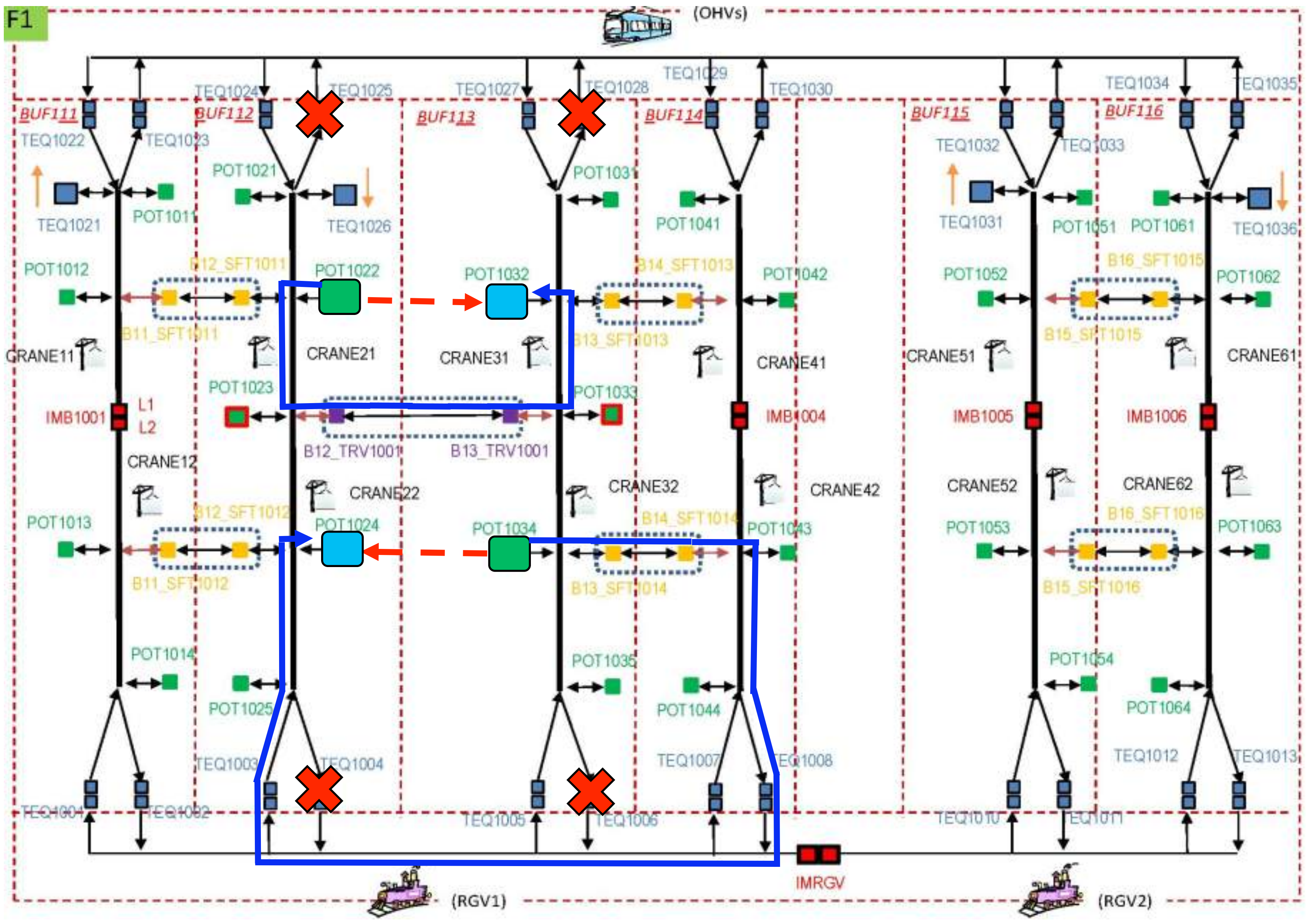
PARC-IHI: LCD Manufacturing Plant





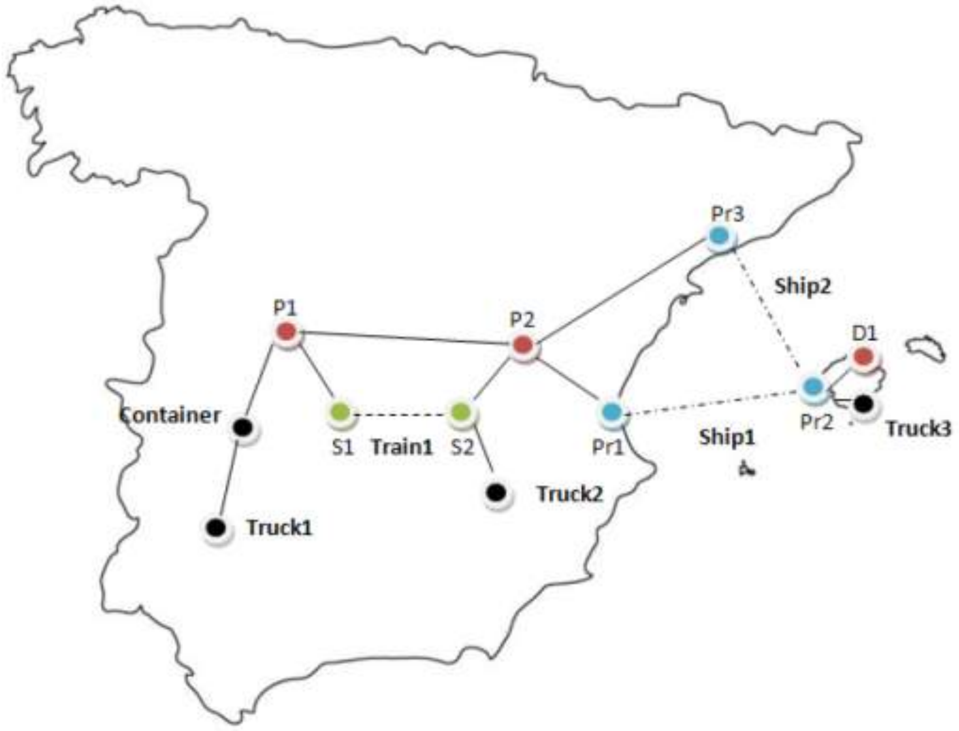
Maximum productivity
Real-time: avoid failures, maintenance, congestion

Deadlock Avoidance: Example 1



Multi-Modal Transportation

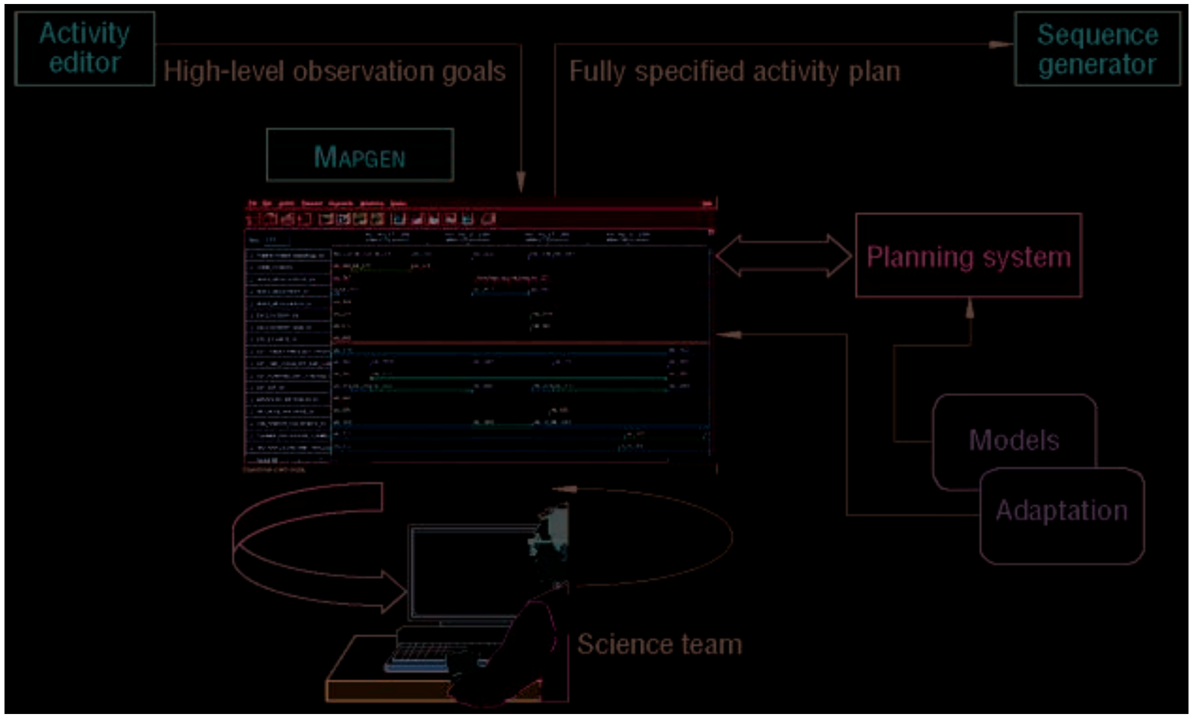
[Borrajo et al., 2009]



600 graph nodes, 179K edges, 300 trucks, 300 containers,
300 transportation routes, 42 train segments, 148 ship segments

Planning time requirement: < 2 hrs

Mapgen: Mixed-Initiative Planning & Scheduling for the Mars Exploration Rover Mission



Using EUROPA Planner developed at NASA Ames
(also used to control: underwater autonomous vehicle at MBARI and robots)

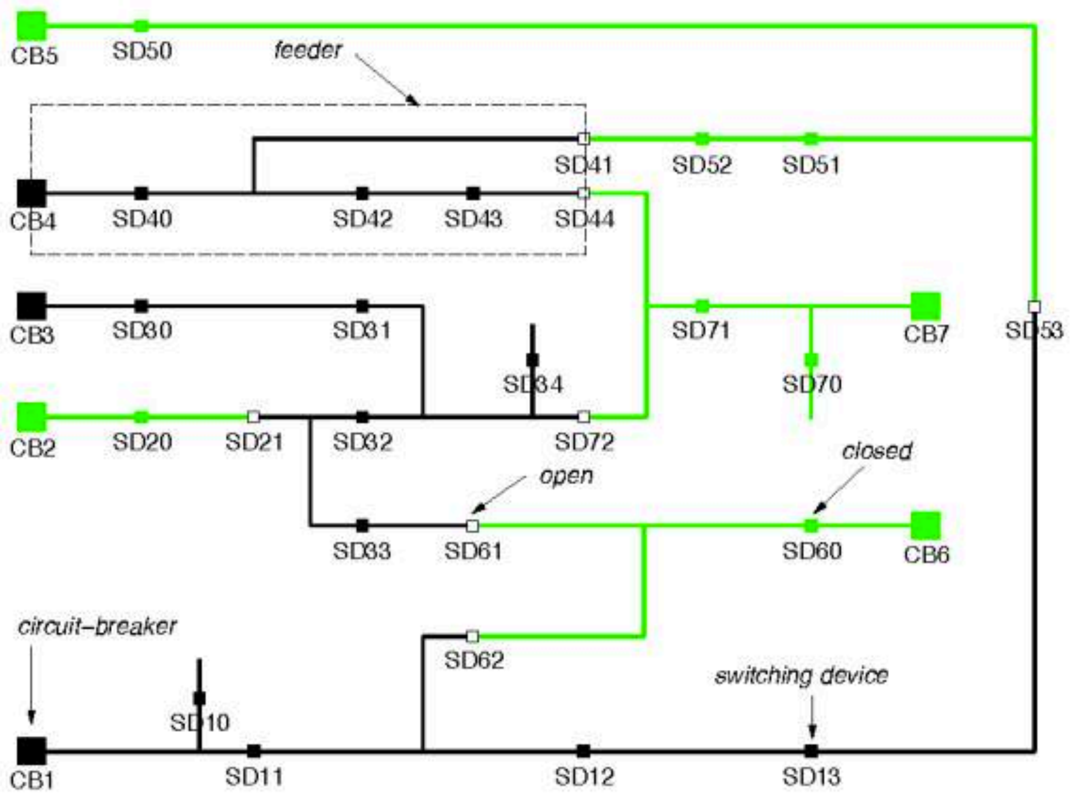
ASPEN@JPL: spacecraft operation, mission design, Antenna utilization, coordinated multiple rover planning

parc
Palo Alto Research Center

Application: Planning with Uncertainty

- Power Supply Restoration (PSR)
- Workflow/Web-service composition
- Interactive Storytelling
- Robot Information Processing & Sensing

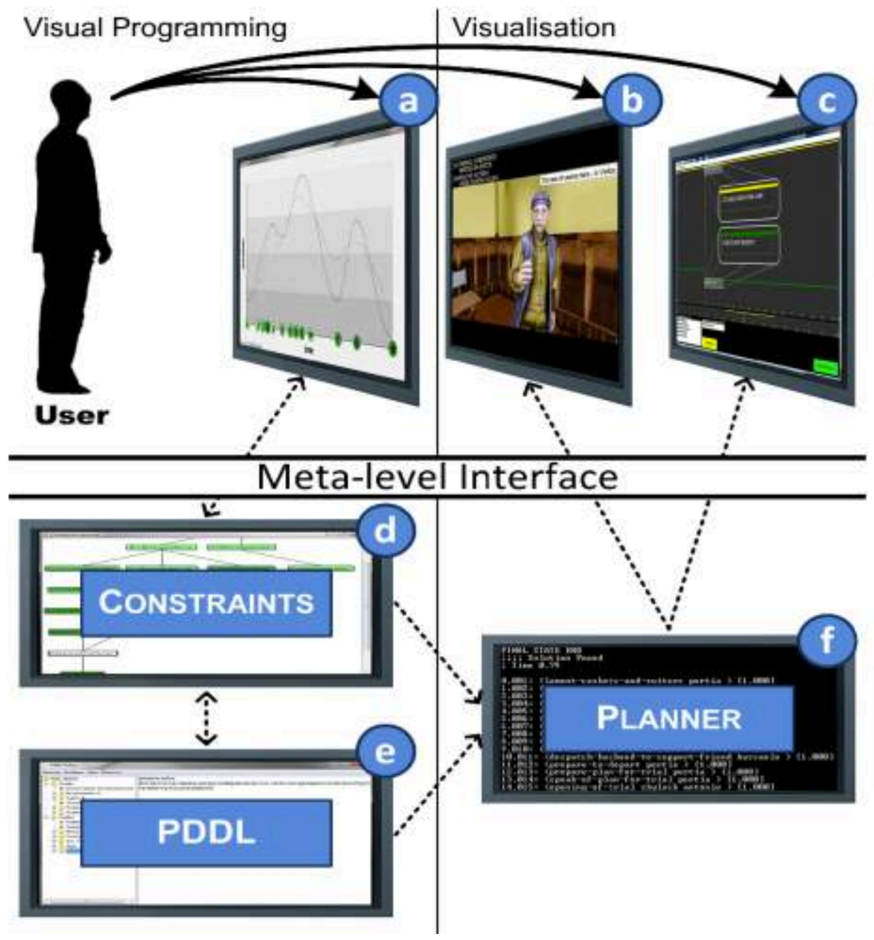
Power Supply Restoration



Supply restoration on faulty power distribution system:
(1) Localize the faulty line; (2) reconfigure the network
→ A natural contingency planning problem

parc Palo Alto Research Center

Interactive Storytelling



Automatically generate story based on user's preferences

Scheduling Applications

- Airport
- Satellite coordination
- Shipping port
- Timetabling
- Google calendar improvements

Conclusion