Basic Concepts & Planning Procedures

Susanne Biundo
University of Ulm
Germany

Artificial Intelligence Planning I

*The field of **AI Planning and Scheduling** is concerned with all aspects of the system-supported synthesis and execution of plans of action.*

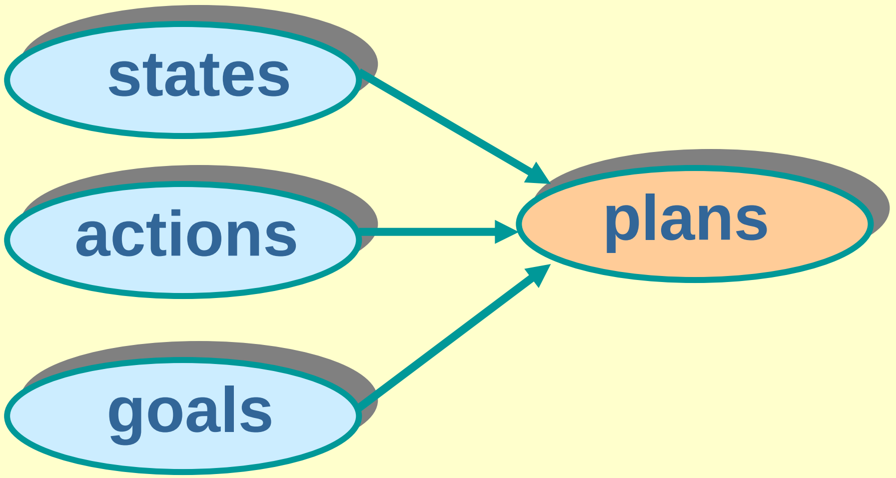
The history of the field dates back to 1967, when Cordell Green presented the first approach.

Artificial Intelligence Planning II

General aim

Enable intelligent goal-directed behaviour

Basic entities

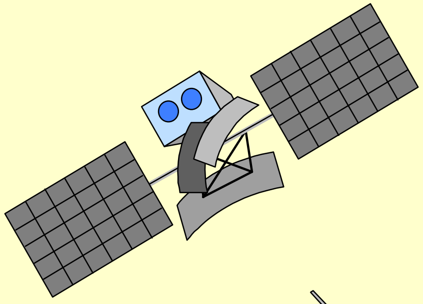


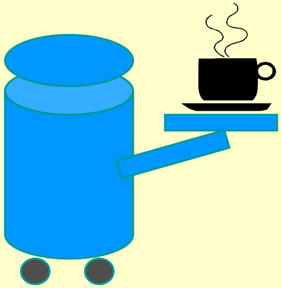
Actions I

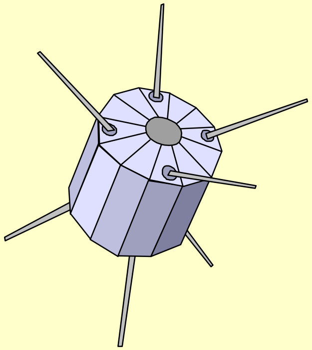
***Actions* perform state changes**

- **actions as to be executed by autonomous systems or virtual agents**

⇨ **system control**







Actions II

- activities as to be pursued in business and production processes

⇒ process control

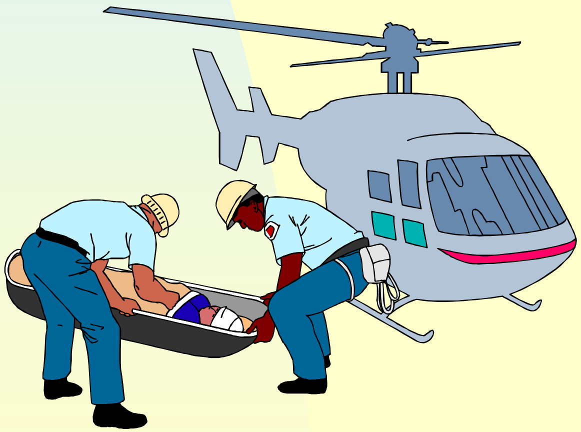
production of physical goods

workflow management

Actions III

- **tasks as to be performed in management and organisation processes**

mission planning



project planning

Artificial Intelligence Planning

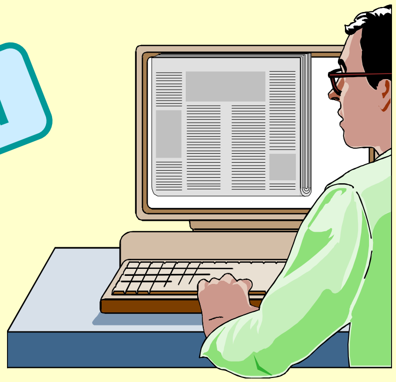
The field of AI Planning and Scheduling is concerned with all aspects of the system-supported synthesis and execution of plans.

They range over a variety of functions.

Dealing with Plans

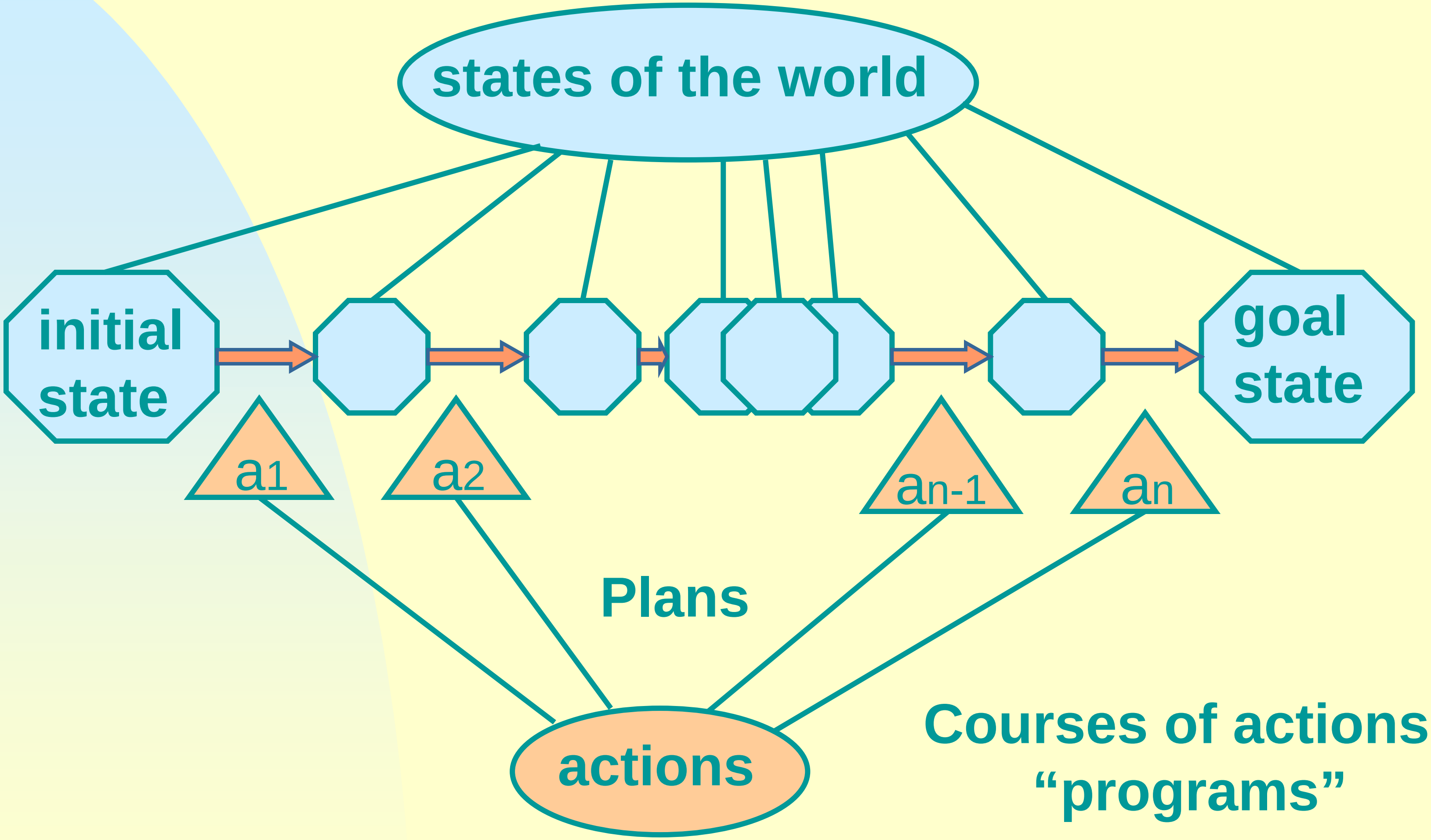
- Synthesis of plans
  - plan generation
- Inspection of plans
  - plan validation
  - plan verification
- Modification of plans
  - plan repair
  - plan merging
- Scheduling

fully automated



interactive

States, Actions, and Plans



Planning vs. Scheduling

- Planning
  - initial state I
  - goal state G
  - operators O
- find a sequence of operator instances from O to transform I into G

⇨ select appropriate actions

⇨ arrange the actions

⇨ consider causalities

- Scheduling
  - a set of actions A
  - a set of resources R
  - a set of constraints C
- find an optimal schedule

⇨ arrange the actions

⇨ assign the resources

⇨ satisfy the constraints

such that the schedule is (close to) optimal

Outline

- Representations of planning domains
  - actions, states, and planning problems
- Classical planning
  - state-space- and plan-space-based approaches
  - graph-based planning
- Heuristic search planning
- Logic-based planning:
  satisfiability testing and deduction
- Hierarchical and hybrid planning
- Historical aspects

Application Domain Characteristics

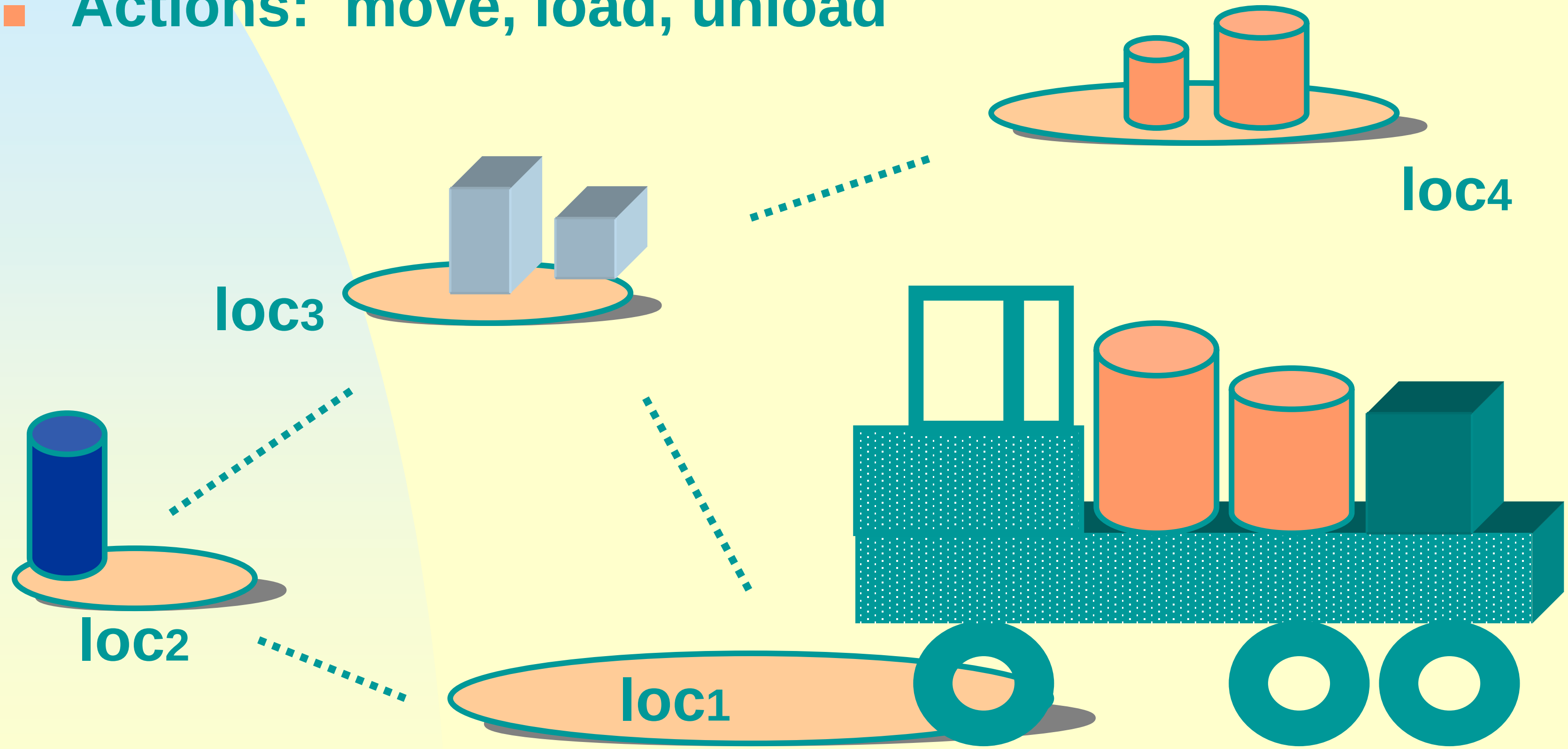
- **static vs. dynamic**
- **finite vs. infinite**
- **deterministic vs. non-deterministic**
- **completely vs. partially observable**
- **hierarchically structured vs. flat**
- **instantaneous vs. durative actions**
- **simple vs. complex plans**
  - **action sequences**
  - **“programs”**
  - **“policies”**

Domain Modelling Requirements

- state and operator descriptions
- planning problems: initial and goal states, intermediate states
- actions as elementary state transitions and elementary constituents of plans
- resources (time, costs, material, personnel)
- domain structure
- type of orderings on actions
- plan structure

An Example Domain

- Objects: trucks, locations, cargo (barrels, boxes)
- Actions: move, load, unload



State Descriptions

adapted from STRIPS (Fikes & Nilsson 1971)

- **first-order language comprising**
  - **a finite set of predicate symbols, like {At, On, Colour, ...}, with arities**
  - **a finite set of constant symbols, like {loc_1, loc_2,...,truck, ba_1, ...}**
  - **an enumerable set of variable symbols, like {x, l, t, ...}**
  - **connectives** $\neg$, $\wedge$
- **states are represented by sets (conjunctions) of ground literals**

States

- states $\sigma \in \Sigma$ are interpretations
- they assign truth values to ground atoms and ground formulae according to
  - $\sigma \models \varphi$ iff $\sigma(\varphi) = \text{true}$, if φ atomic
  - $\sigma \models \neg \varphi$ iff not $\sigma \models \varphi$
  - $\sigma \models \varphi_1 \,\&\, \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$

Operator Descriptions

- **Operators have three components**
  - **an operator symbol $o \in O$ (name) of arity n and a list of terms t1, ..., tn (arguments), which occur in the preconditions and effects of the operator**
  - **a set (conjunction) of literals prec(o), the preconditions of the operator**
  - **a set (conjunction) of literals eff(o), the effects of the operator**

Operator Application I

- **effects of operators are often described by two disjoint sets**
  - **add(o) = $\{l \mid l \in eff(o)$, l atom$\}$ (add list)**
  - **del(o) = $\{l \mid \neg l \in eff(o)\}$ (delete list)**
- **a set S of ground literals describes a set of states**
- **a ground instance a of an operator, i.e. an action, is applicable to S iff**
  - **prec(a) $\subseteq$ S**

Operator Application II

- **the application of a to S results in a set of literals**

  $$T = (S - del(a) - \{\neg\, l \mid l \in add(a)\}) \cup eff(a)$$

- **T describes the set of possible successor states to S w.r.t a (T = succ (S, a))**

- **simplification**
  - **CWA : S contains only atoms**
  - **prec(a) contains only atoms**
  - **$T = succ(S, a) = (S - del(a)) \cup add(a)$, if $prec(a) \subseteq S$**

Planning Problems

- a planning problem (plan specification)

  P = (init, goal, O)

  - init and goal : sets of literals describing the initial and goal states of the problem
  - O : set of operators

- a sequence $a_1 \ldots a_m$ of operator ground instances (actions) is a **plan** iff
  - a_1 is applicable to some state description S
  - a_i is applicable to each state description resulting from the application of a_{i-1} ($1 < i \leq m$) to the resp. predecessor

Solutions to Planning Problems

- **a sequence $a_1 \dots a_m$ of actions is a solution to a planning problem P = (init, goal, O) iff**
  - **$a_1 \dots a_m$ is a plan**
  - **a_1 is applicable to init**
  - **each goal-literal is in the successor state description of a_m**

- **$a_1 \dots a_m$ is a total-order plan**

Semantic Properties

- an action is **sound** iff there exist states $\sigma, \sigma' \in \Sigma$ such that $\sigma \models prec(a)$ and $\sigma' \models eff(a)$
- a sound action specifies a **state transition**
- if an action a is applicable to a state description S and $\sigma \models S$ for a state $\sigma \in \Sigma$, then $\sigma \models prec(a)$
- if $\sigma \models S$ and a is applicable to S and T is the successor state description to S w.r.t. a, then there exists $\sigma' \in \Sigma$ with $\sigma' \models T$

Correctness of Plans

- **a plan that is a solution to a planning problem P = (init, goal, O) specifies a set of state transitions from each state satisfying init to a state satisfying goal**
- **such a plan is correct w.r.t. the given problem (plan specification)**

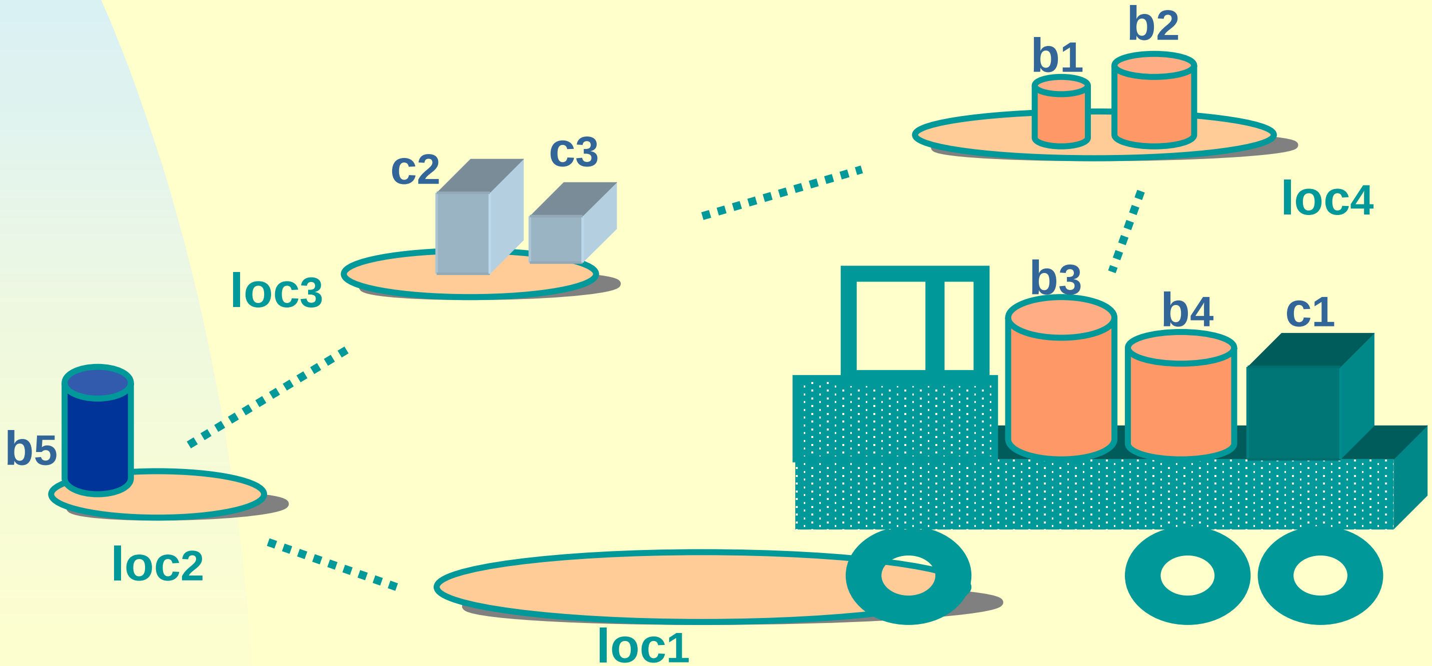
Example I

State description

At (truck, loc1), On (truck, b3), On(truck, b4) , ...,

At (truck, loc1), At (b3, loc1), ..., At (b1, loc4), ...,

Connected (loc1, loc2), ...



Example II

Operator descriptions

- **move (t: truck, l_1, l_2 : location)**
  prec: At (t, l_1), Connected (l_1, l_2)
  add: At (t, l_2)
  del: At (t, l_1)

- **unload (t: truck , l: location, c: cargo)**
  prec: At (t, l), On (t, c)
  add: At (c, l)
  del: On (t, c)

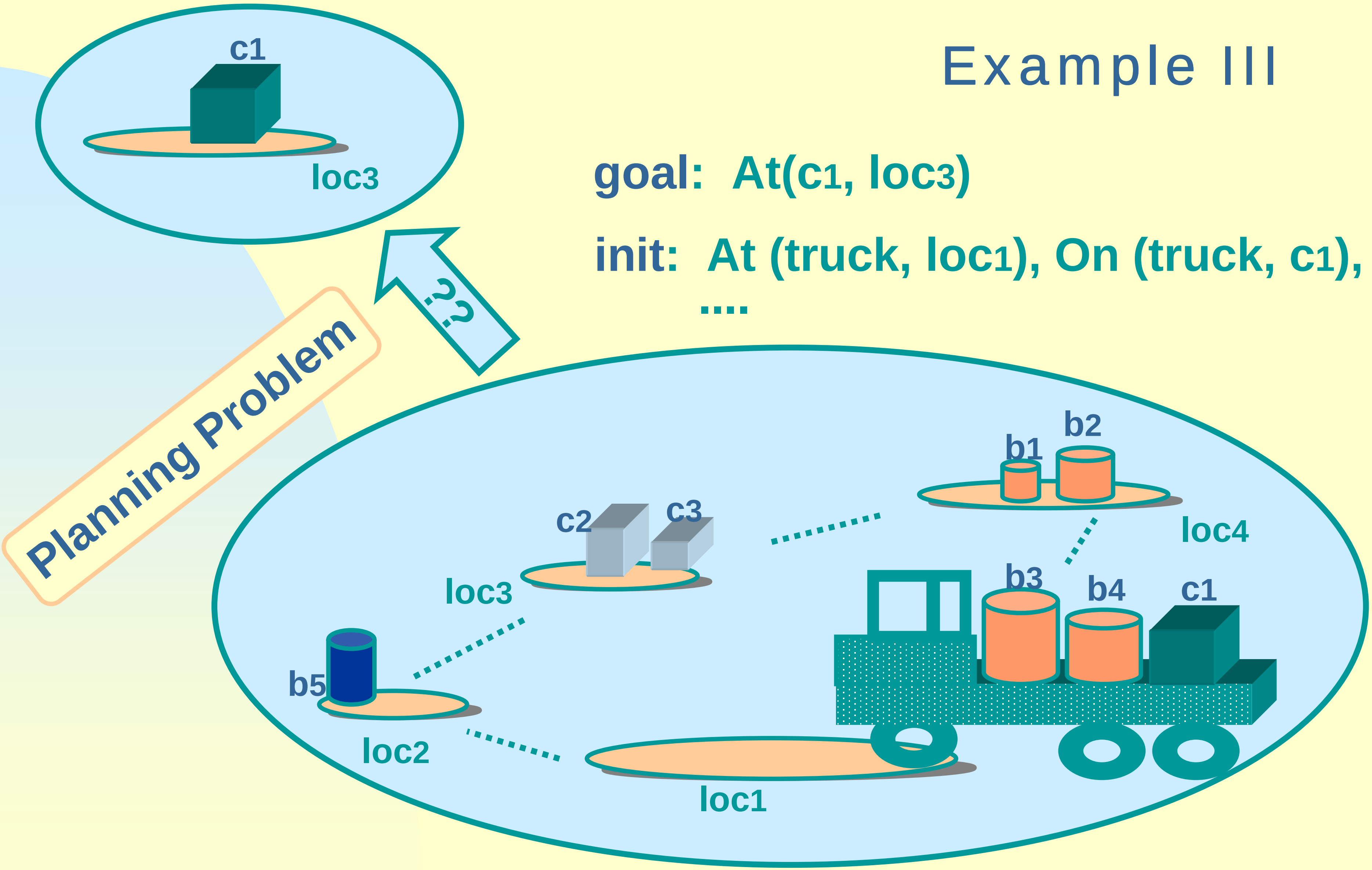
- **load (t: truck, ...)**
  prec: ...

Example III

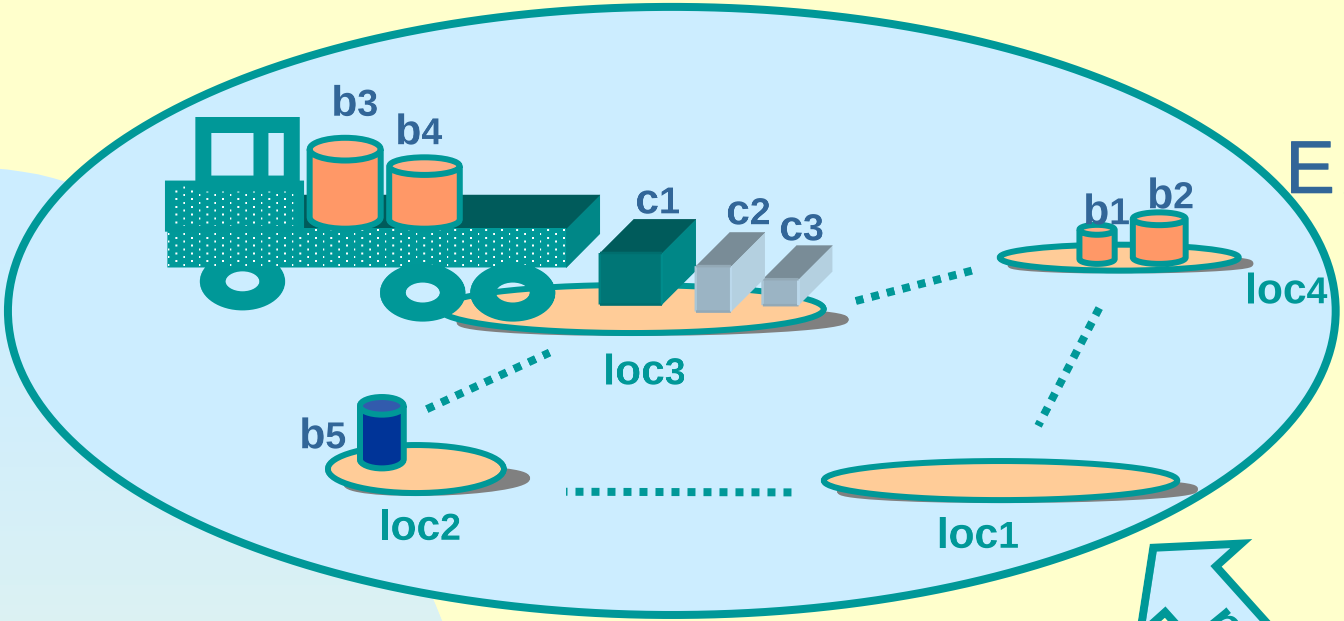
goal: At(c_1, loc_3)

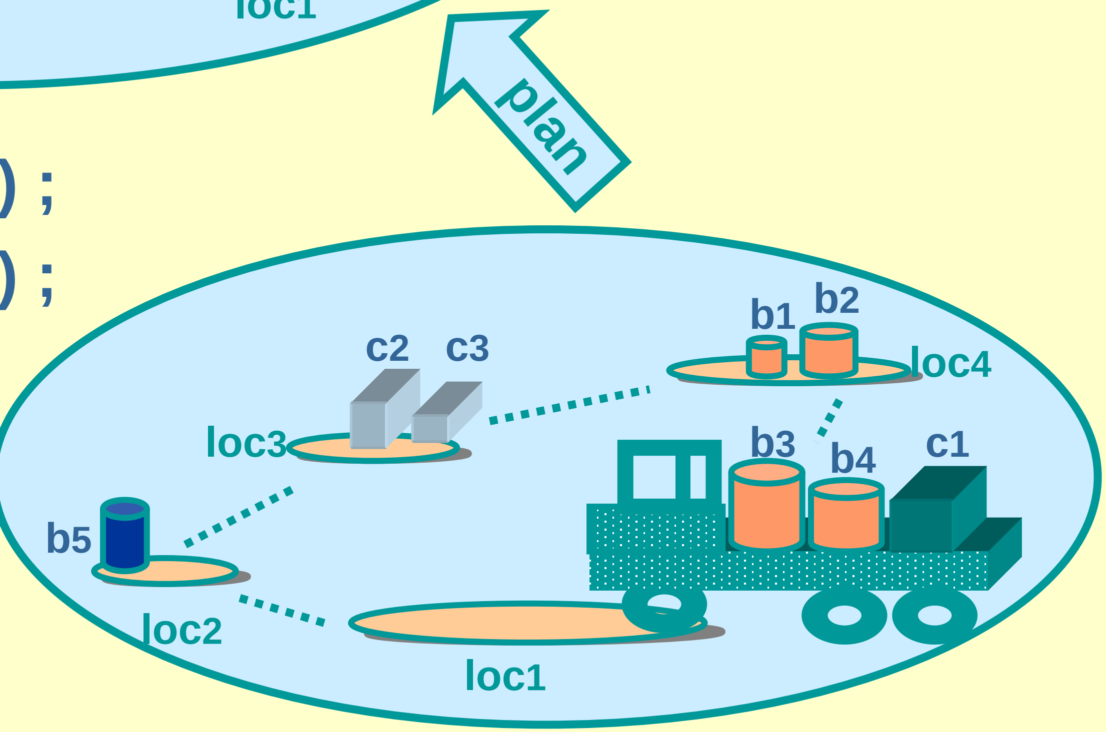
init: At (truck, loc_1), On (truck, c_1),

....



Example IV





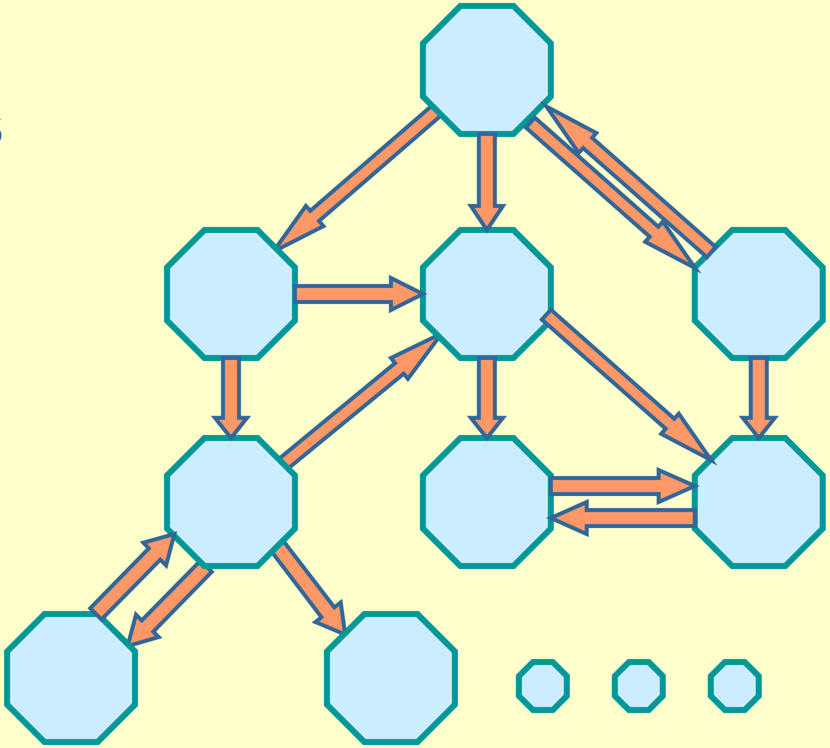
move (truck, loc1, loc2) ;

move (truck, loc2, loc3) ;

unload (truck, loc3, c1)

How to find a plan

- state-space search
  - search space ⟹ subset of state space
  - nodes ⟹ states
  - edges ⟹ actions



- find a path from the initial state to a goal state

Forward Search

progression-planning (init, goal, O)

```
S := init
p := ε
repeat
   if  goal ⊆ S  then  return p
   A := { a | prec(a) ⊆ S  and
              a ground instance of some o ∈ O}
   if  A = ∅  then  fail
   choose  a ∈ A
   S := succ (S, a)
   p := a;p
```

Backward Search

regression-planning (init, goal, O)

S := goal
p := ε
repeat
  if S ⊆ init **then return** *p*
  A := { a | (eff(a) ∩ S) ≠ ∅ and
      a ground instance of some o ∈ O}
  if A = ∅ **then fail**
  choose a ∈ A
  S := wp (S, a) = (S ⊖ eff(a)) ∪ prec(a)
  p := *p*;a

Guiding the Search

How to implement efficient planning procedures ?

- ⇨ **reduction of the search space**
- ⇨ **structuring the search space**
- ⇨ **goal-directed procedures**
- ⇨ **strategies**
- ⇨ **heuristics**
- ⇨ **encodings**

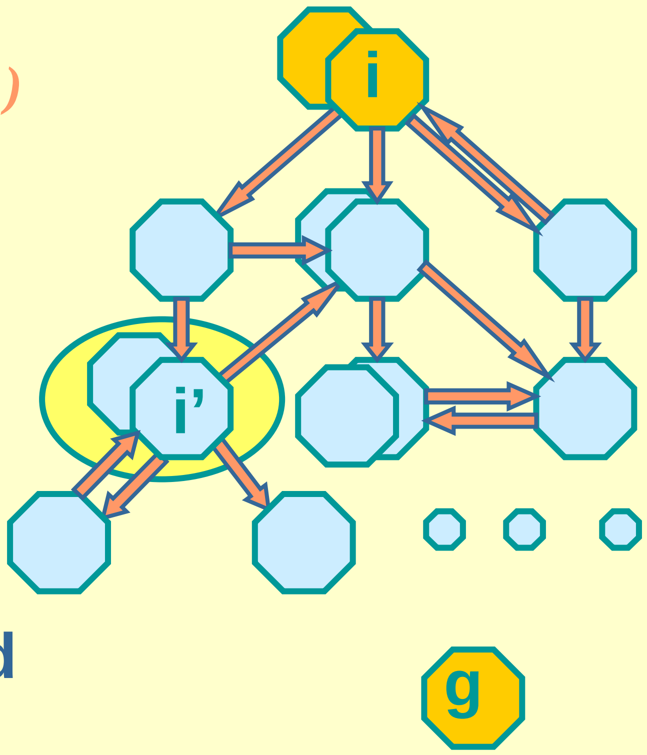
Example Strategy

island search

means ends analysis

STRIPS - Algorithm (Fikes & Nilsson 1971)

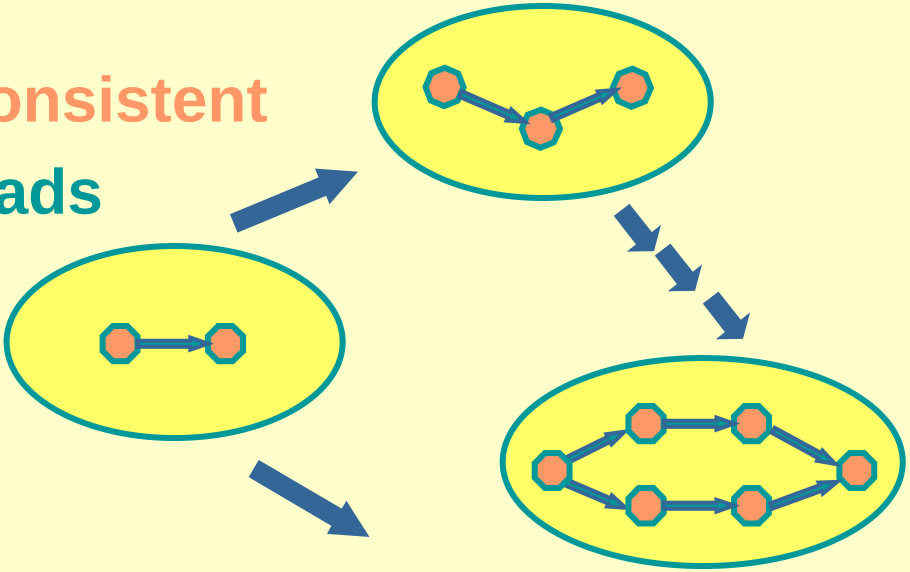
- select a goal
- achieve the goal from initial state
- compute new initial state
- select next goal
- repeat until all goals achieved



How to find a plan

- **plan-space search**
  - **search space ⟹ space of partial plans**
  - **nodes ⟹ partial plans**
  - **edges ⟹ plan refinement operations**

- **find a complete and consistent non-linear plan that leads from the initial state to a goal state**



Non-linear Plans I

np = (PlanSteps, OrderingConstraints, CausalLinks, VariableBindings, ...)

- **PlanSteps** : a set of labeled operations
- **OrderingConstraints** : a partial order on PlanSteps
- **CausalLinks** : a set of causal relations between plan steps

  ($ps_i \xrightarrow{c} ps_j$) : ps_i establishes the precondition c of ps_j
- **VariableBindings** : $x = t$, x variable, t term

Non-linear Plans II

a non-linear plan *np* solves a planning problem $P = (\mathrm{init}, \mathrm{goal}, O)$ iff *np* is a complete and consistent refinement of the initial plan

$$(\{ps_\alpha, ps_\omega\}, \{(ps_\alpha < ps_\omega)\}, \emptyset, \emptyset)$$

ps_α and ps_ω are fictive plan steps with

$$\mathrm{eff}(ps_\alpha) = \mathrm{init}$$

$$\mathrm{prec}(ps_\omega) = \mathrm{goal}$$

Non-linear Plans III

- *np* is complete iff
  - each precondition of each plan step is established by a plan step
  - each linearisation of *np* is safe, i.e. no step ps_k with $c \in del(ps_k)$ can occur between ps_i and ps_j, if ps_i establishes a precondition c of ps_j

- *np* is consistent iff
  - its ordering constraints are consistent,
    i.e. if $ps_i < ps_j$ then not $ps_j < ps_i$
  - its variable bindings are consistent,
    i.e. if $x = A$ then not $x = B$, if $A \neq B$

Example

At(c1, loc3)

At(b5, loc4)

ps_ω

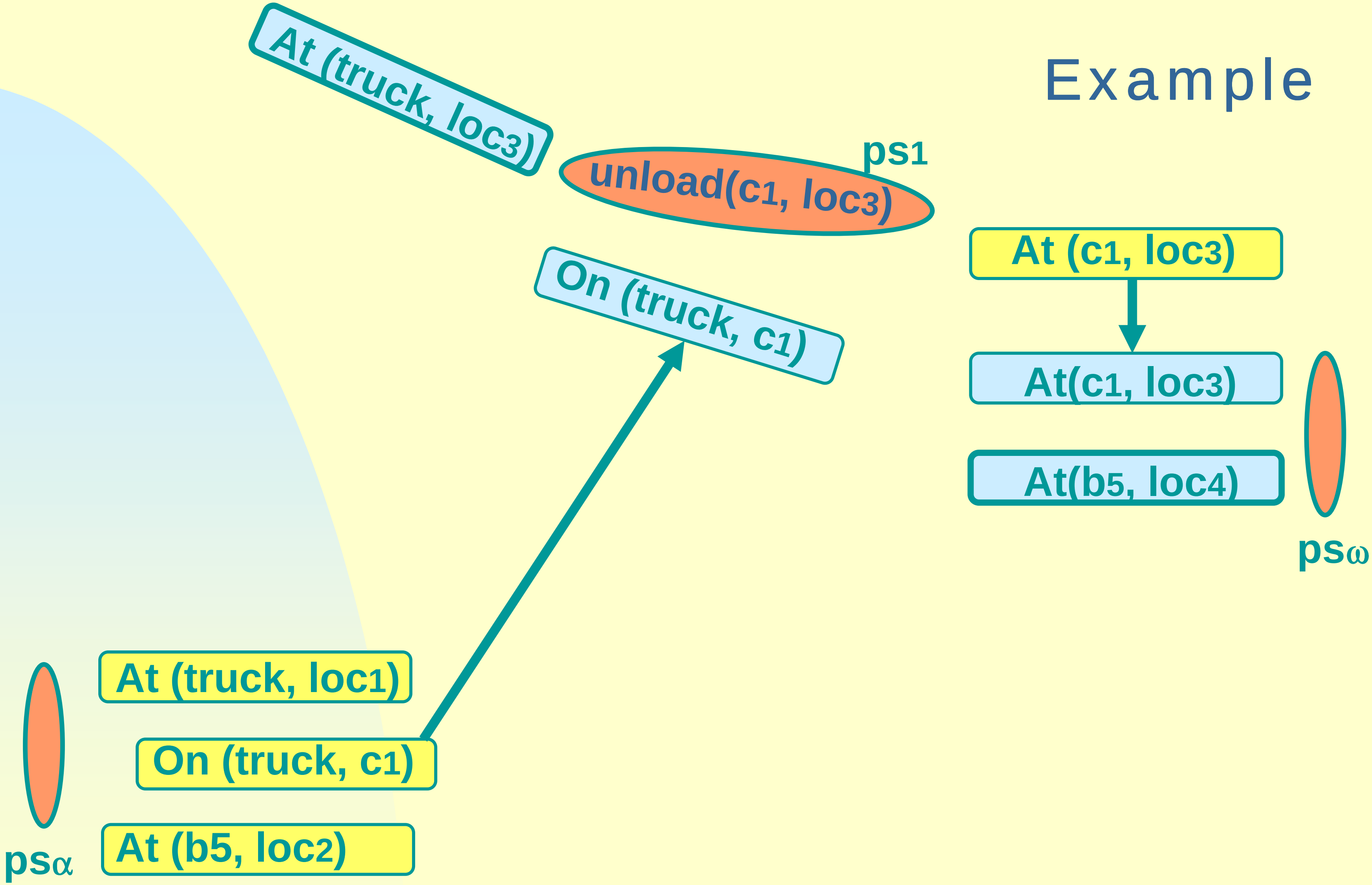
At (truck, loc1)

On (truck, c1)

At (b5, loc2)

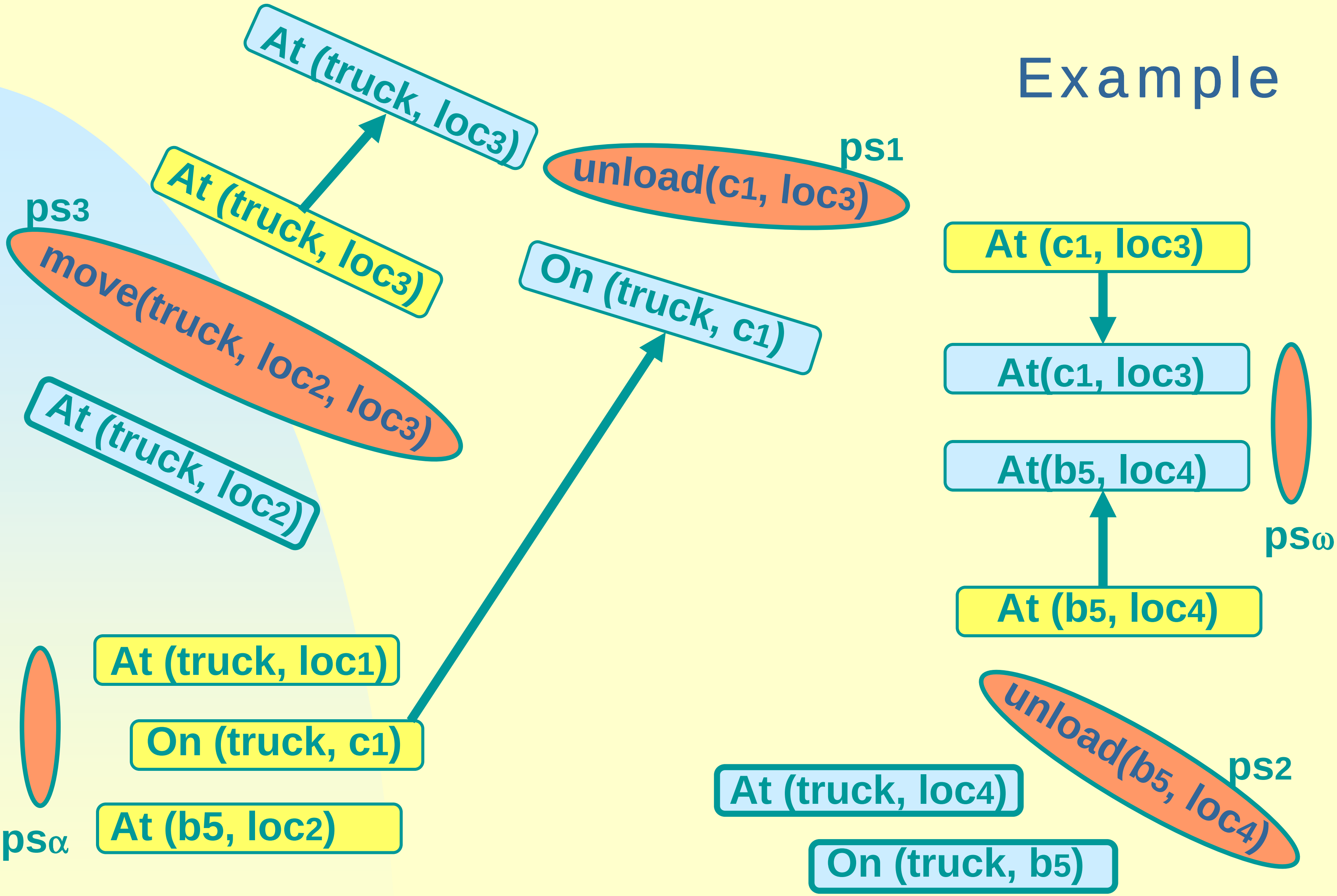
ps_α

Example

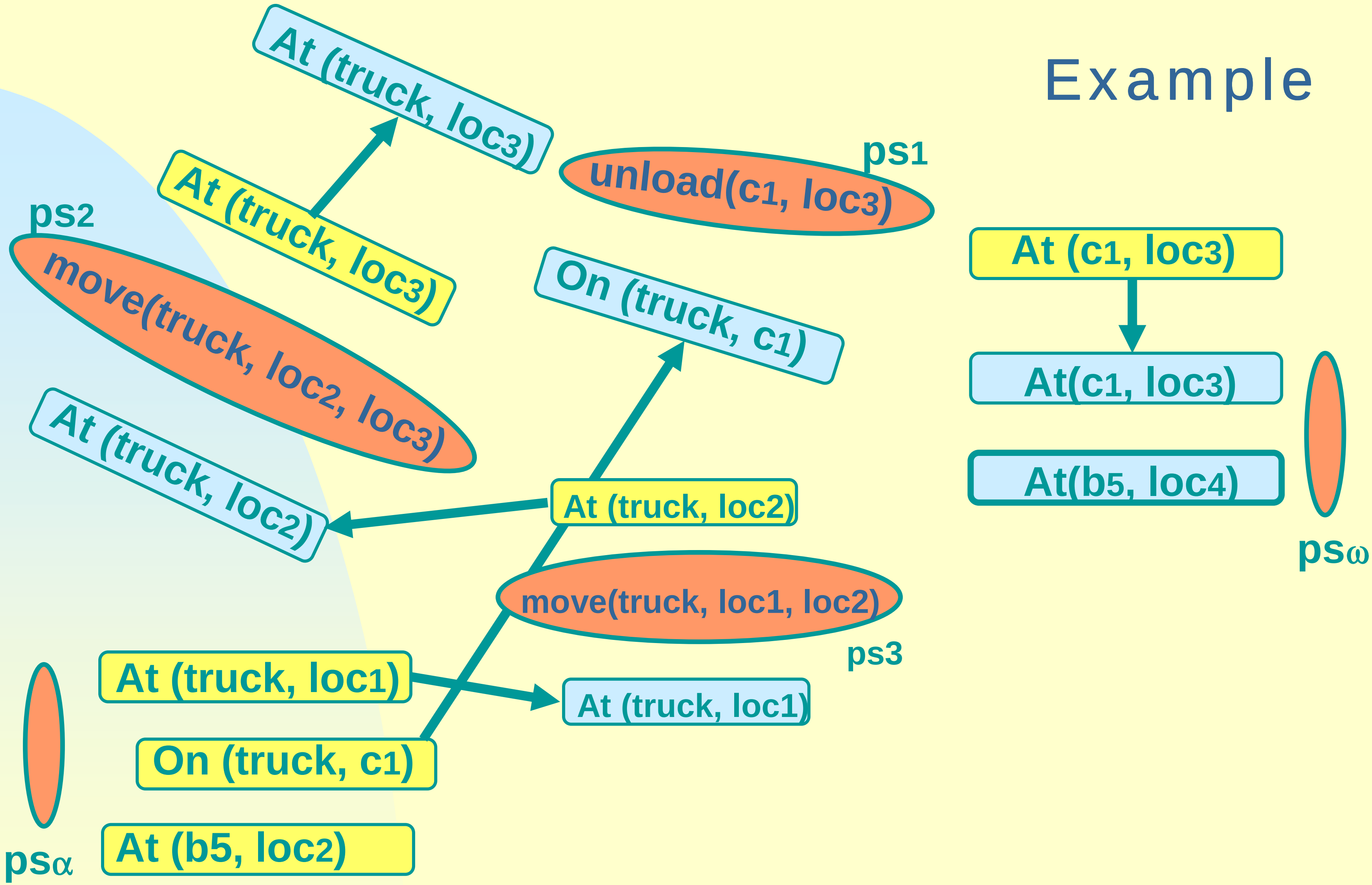


At (truck, loc_3)
ps_1
unload(c_1, loc_3)
At (c_1, loc_3)
On (truck, c_1)
At(c_1, loc_3)
At(b_5, loc_4)
ps_ω
At (truck, loc_1)
On (truck, c_1)
ps_α
At (b5, loc_2)

Example

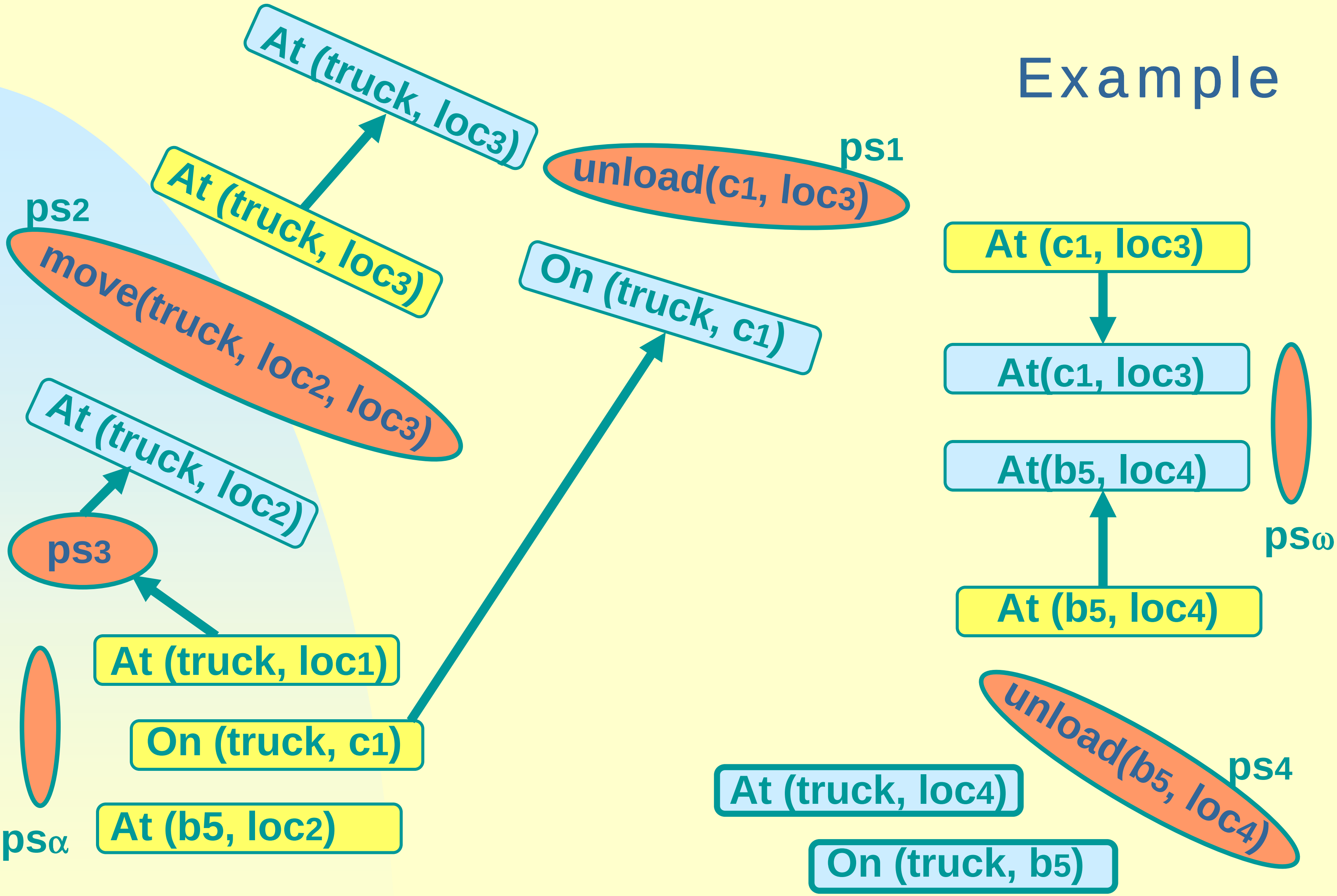

At (truck, loc3)
At (truck, loc3)
ps3
move(truck, loc2, loc3)
At (truck, loc2)
ps1
unload(c1, loc3)
On (truck, c1)
At (c1, loc3)
At(c1, loc3)
At(b5, loc4)
psω
At (b5, loc4)
At (truck, loc1)
On (truck, c1)
At (b5, loc2)
psα
unload(b5, loc4)
ps2
At (truck, loc4)
On (truck, b5)

Example



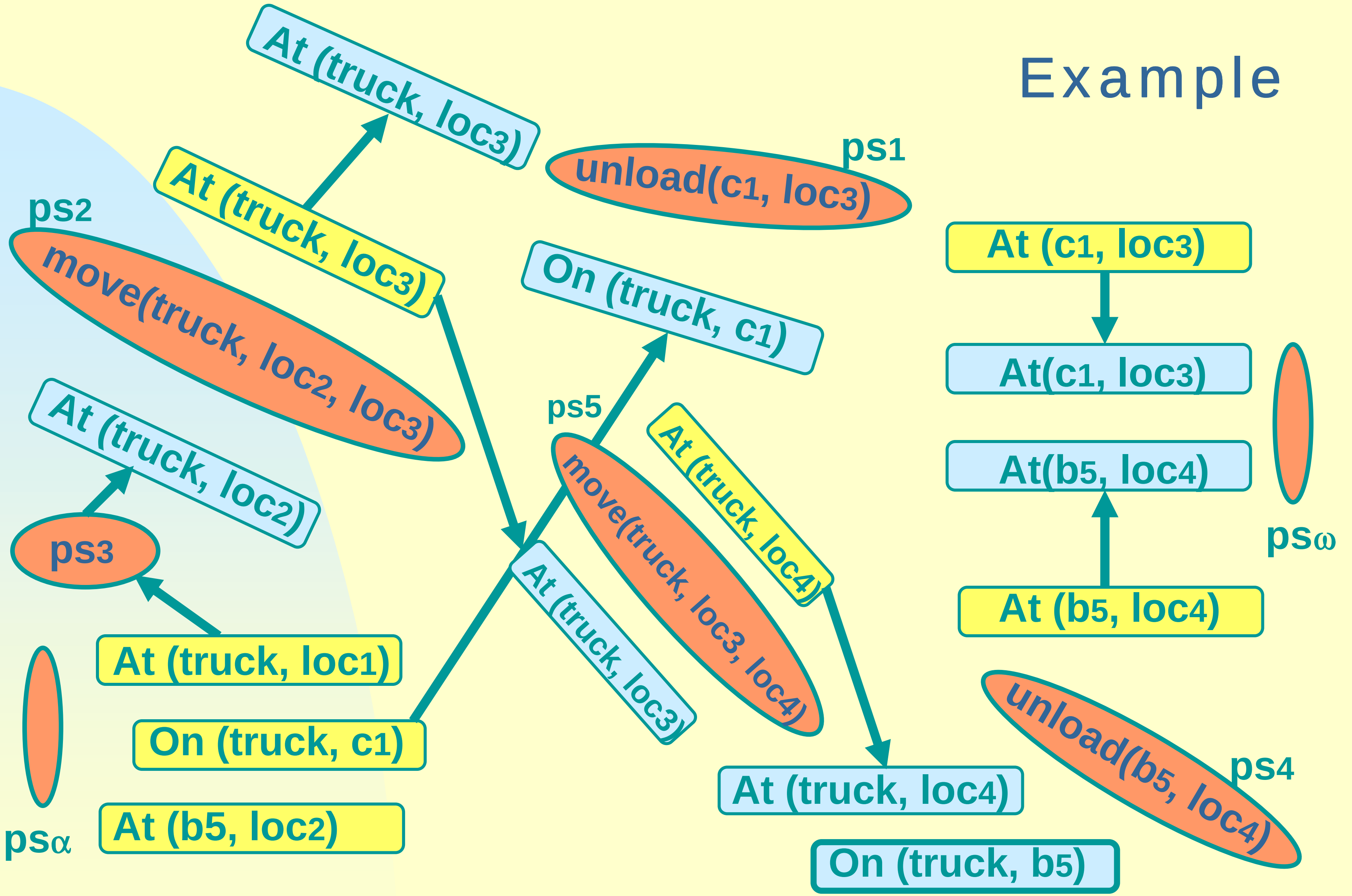
At (truck, loc3)
ps1
unload(c1, loc3)
ps2
At (truck, loc3)
move(truck, loc2, loc3)
On (truck, c1)
At (c1, loc3)
At(c1, loc3)
At(b5, loc4)
psω
At (truck, loc2)
At (truck, loc2)
move(truck, loc1, loc2)
ps3
At (truck, loc1)
At (truck, loc1)
On (truck, c1)
At (b5, loc2)
psα

Example

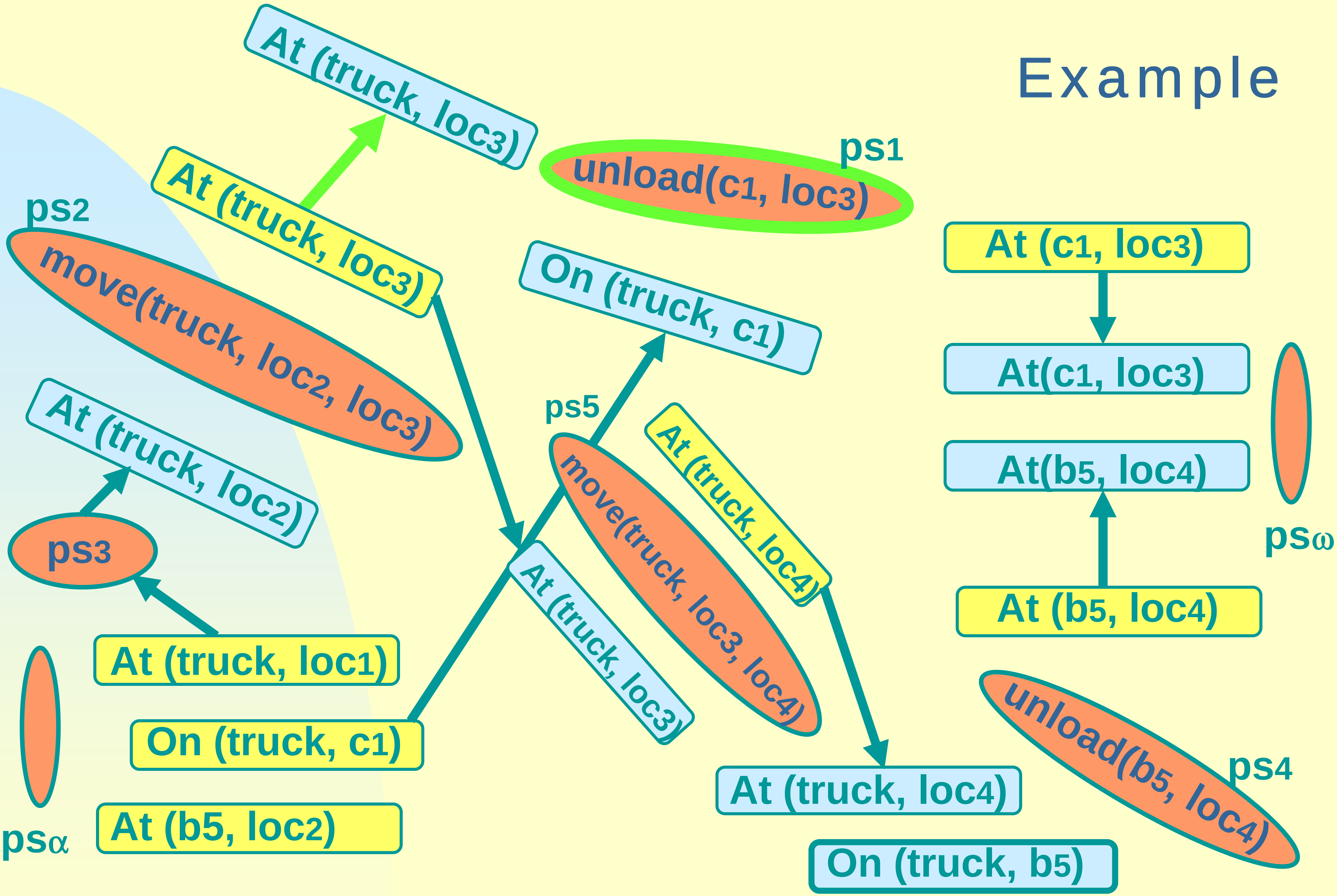


At (truck, loc3)
ps1
unload(c1, loc3)
At (truck, loc3)
ps2
move(truck, loc2, loc3)
On (truck, c1)
At (c1, loc3)
At(c1, loc3)
At (truck, loc2)
At(b5, loc4)
psω
ps3
At (b5, loc4)
At (truck, loc1)
On (truck, c1)
unload(b5, loc4)
ps4
At (truck, loc4)
psα
At (b5, loc2)
On (truck, b5)

Example


At (truck, loc3)
ps1
unload(c1, loc3)
At (truck, loc3)
ps2
move(truck, loc2, loc3)
On (truck, c1)
At (c1, loc3)
At(c1, loc3)
At(b5, loc4)
At (b5, loc4)
psω
ps5
At (truck, loc4)
move(truck, loc3, loc4)
At (truck, loc2)
ps3
At (truck, loc3)
At (truck, loc1)
On (truck, c1)
At (b5, loc2)
psα
At (truck, loc4)
On (truck, b5)
unload(b5, loc4)
ps4

Example



At (truck, loc3)
ps1
unload(c1, loc3)
ps2
At (truck, loc3)
move(truck, loc2, loc3)
At (c1, loc3)
On (truck, c1)
At(c1, loc3)
ps5
At (truck, loc2)
At (truck, loc4)
At(b5, loc4)
move(truck, loc3, loc4)
psω
ps3
At (truck, loc3)
At (b5, loc4)
At (truck, loc1)
On (truck, c1)
unload(b5, loc4)
ps4
At (truck, loc4)
psα
At (b5, loc2)
On (truck, b5)

Partial-Order Causal-Link Planning

SNLP - Algorithm (McAllister & Rosenblitt 1991)

np := initial-plan(init, goal, O)

pocl (np)

if np inconsistent **then return fail**

if np complete **then return** np

if there is a causal link ($ps_i \xrightarrow{c} ps_j$)

"threatened " by a step ps_k , i.e. $c \in del(ps_k)$

and ps_k possibly inbetween ps_i and ps_j

then choose

Promotion : return pocl ($np \oplus (ps_k < ps_i)$)

Demotion : return pocl ($np \oplus (ps_j < ps_k)$)

Partial-Order Causal-Link Planning ctd.

<u>**choose**</u> ps_k and $c \in prec(ps_k)$ where there is no causal link $(ps_l \xrightarrow{c} ps_k)$ for some ps_l

<u>**choose**</u>

a. <u>**choose**</u> ps_m from np such that $c \in eff(ps_m)$
return pocl $(np \oplus (ps_m \xrightarrow{c} ps_k))$

b. <u>**choose**</u> an operator $o \in O$ and a variable instantiation δ such that $c \in eff(\delta(o))$
generate a new plan step ps_n from $\delta(o)$
np := insert (np, ps_n)
return pocl $(np \oplus (ps_n \xrightarrow{c} ps_k))$

Plan-space vs. State-space Planning

- **plan-space-based planning**
  least commitment principle
  - **generation of non-linear plans**
    ⟹ **partial order on actions**
  - **least restricting variable bindings**
  - **step-wise plan refinement**

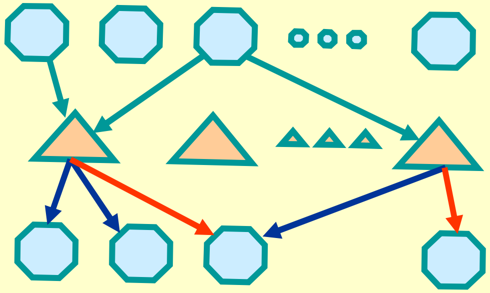
- **state-space-based planning**
  - **generation of linear plans**
    ⟹ **total order on actions**
  - **step-wise plan extension**

Least commitment w.r.t. variable instantiation

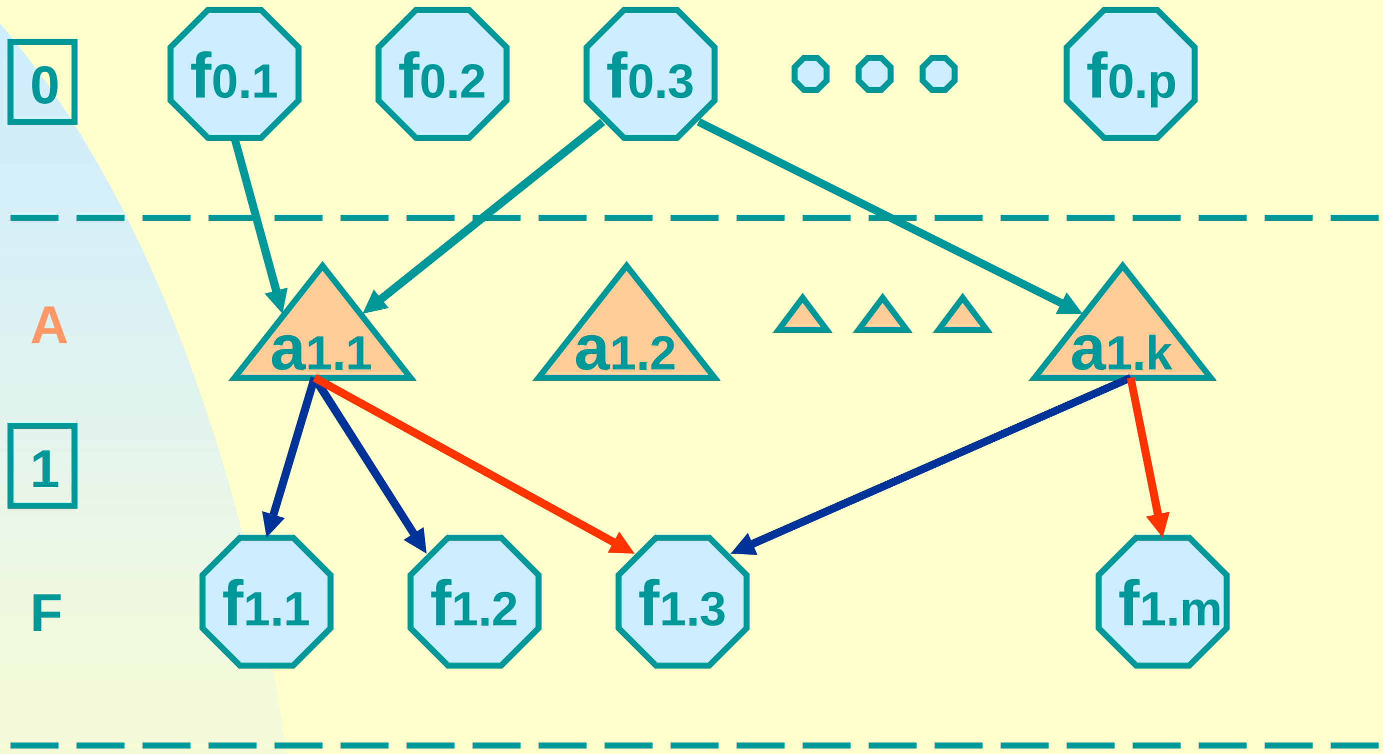
How to find a plan

- **planning-graph construction**
  - **search space ⟹ compact representation of the state space**
  - **nodes ⟹ atoms, actions**
  - **edges ⟹ precondition-, add-, delete-relations**

- **find a sequence of sets of actions that represent a complete and consistent partial order plan leading from the initial state to a goal state**



Planning-Graphs



F0 : init **Fn : goal**

Graph Construction and Analysis I

GRAPHPLAN (Blum & Furst 1995)

two phases

- **construction of the planning graph : forward search**
  - **generate successive levels of the planning graph, starting from the initial state, until a level *n* is reached, which contains the goals**

- **analysis of the planning graph : backward search**
  - **construction of a plan**

Graph Construction and Analysis II

- **collect in level *n* a set of non-mutex actions that achieve the goals**
- **collect in each level *i* a set of independent actions that establish the preconditions of actions collected in level *i+1*   (1 < i < n)**
- **F0 (init) contains the preconditions of actions collected in level 1**

- **two actions are mutual exclusive (mutex) iff they interact or their preconditions are mutex, i.e. are established by non-mutex actions**

Graph Construction and Analysis III

- **two actions a and b interact iff**
  $(prec(a) \cup add(a)) \cap del(b) \neq \emptyset$ or
  $(prec(b) \cup add(b)) \cap del(a) \neq \emptyset$

Properties

- graph-based planning approaches rely on ground instances of state descriptions and operators
- they combine aspects of linear and non-linear planning
- they generate shortest plans in terms of "time steps"
- they always terminate

Heuristic Search Planning I

HSP (Bonet & Geffner 1997)

- **heuristic search**
  - **state-based search (forward, backward)**
  - **heuristic function**
    - **estimate the distance between the current and a goal state resp. the initial state**
    - **optimal cost function of the relaxed planning problem**
      e.g.: ignore the delete lists of operators
  - **A*, IDA*, hill climbing**

Heuristic Search Planning II

- graph-based planning as heuristic search

AltAlt (Long & Kambhampati 2000)

  - backward search
  - heuristic functions are extracted from the planning graph
    - level that contains goals without mutex
    - consider mutex relations between actions

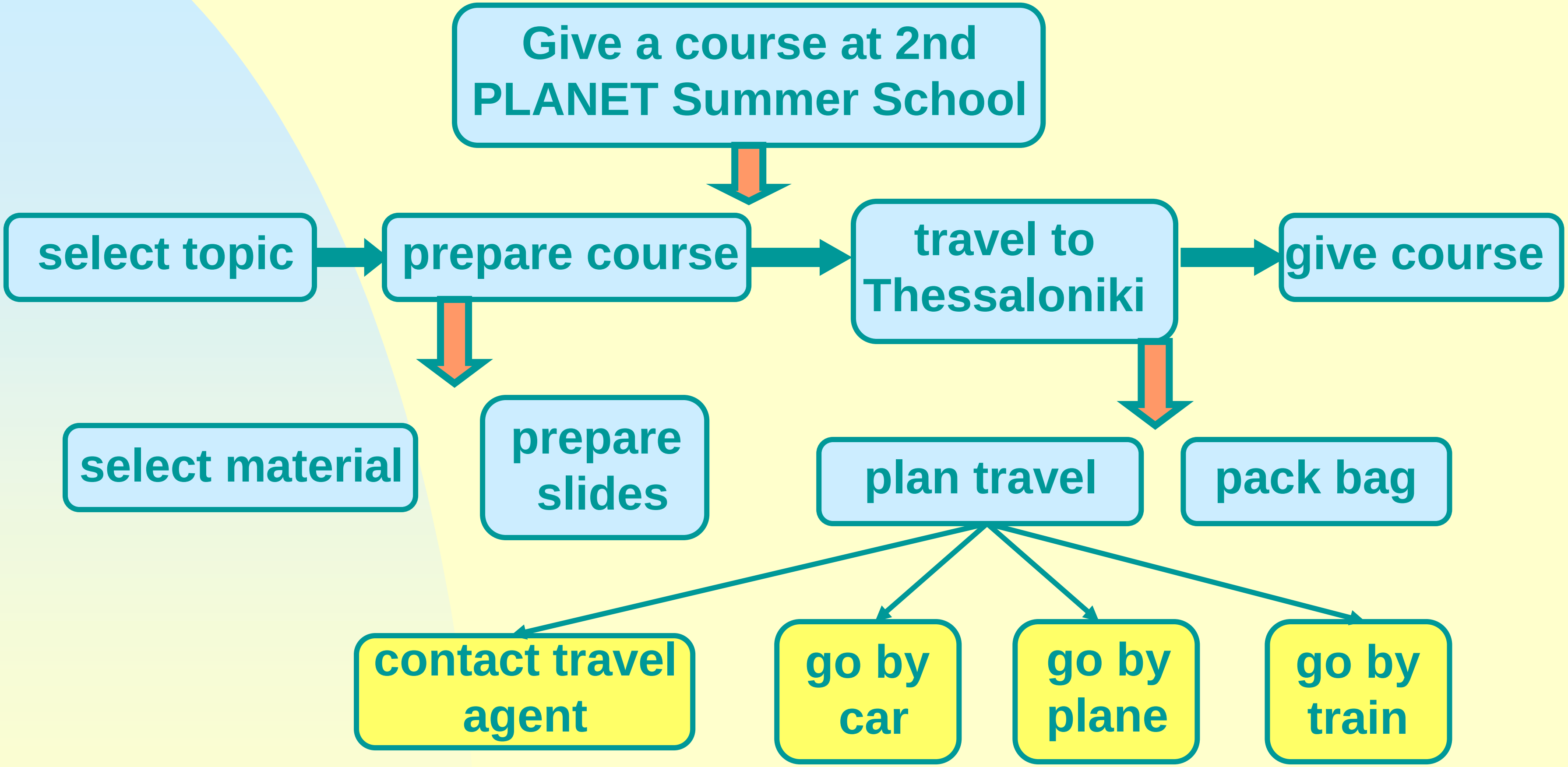
Extended Representations

ADL (Pednault 1986)

- **conditional operators**
- **universally quantified effects**
- **disjunctive preconditions**
- **function expressions in operator descriptions**
  - **state transitions through variable assignments**
  - **similarity to the state variable concept**

simplifying domain descriptions

A Different View on Planning



Hierarchical Planning I

aim solving complex, realistic planning problems

- project planning
- mission planning

requirements

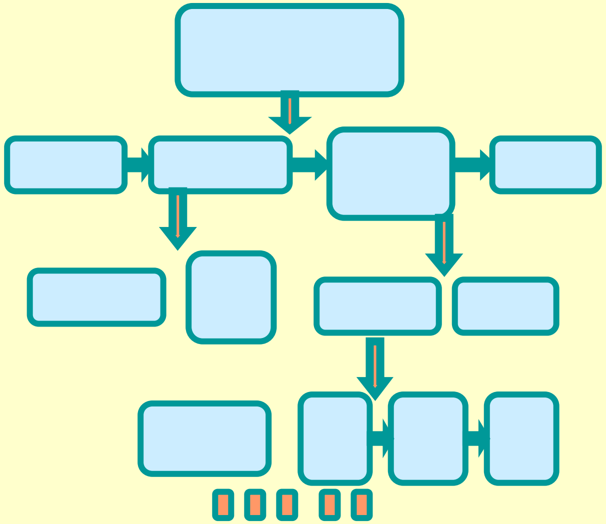
- flexible plan generation
- understandable solutions
- user interaction and predefined user plans

Hierarchical Planning II

- exploiting the inherent hierarchical structure of application domains
- relying on pre-defined, pre-structured, abstract solutions
  - procedural and declarative specifications of abstract operators
- considering ordering and causal constraints

How to find a plan

- **task net construction**
  - **search space ⟹ the space of task networks**
  - **nodes ⟹ task networks**
  - **edges ⟹ refinement and decomposition operations**
- **find a complete and consistent primitive task network that represents a solution of the initial problem (goal task)**



Hierarchical Task Network Planning

O-Plan (Drabble &Tate 1977)   UMCP (Erol, Hendler, Nau 1994)

- **abstract tasks, compound tasks**
- **goal tasks**
- **primitive tasks**
- **decomposition methods : (*t : task*, *tn : task-network*)**

→ **tn : abstract / primitive task network**

→ **tn : pre-defined (abstract) partial plan**

Decomposition Planning I

stepwise expansion of abstract task networks through decomposition operations

- **start from the initial goal task**
- **select and apply appropriate decomposition methods**
- **until a primitive task network is obtained**

Decomposition Planning II

- **consistence and completeness of the primitive task network are guaranteed through the use of correct methods**

- **a method (*t*, *tn*) is correct iff *tn* is a correct implementation of *t***
  simplest case:
  - ***tn* is a complete and consistent (abstract) plan**
  - **the preconditions of tasks in *tn* are established through the preconditions of *t* or within *tn***
  - ***tn* produces the effects of *t***

Application Example: Mission Planning

Crisis management support

THW mission at the river Oder in 1997

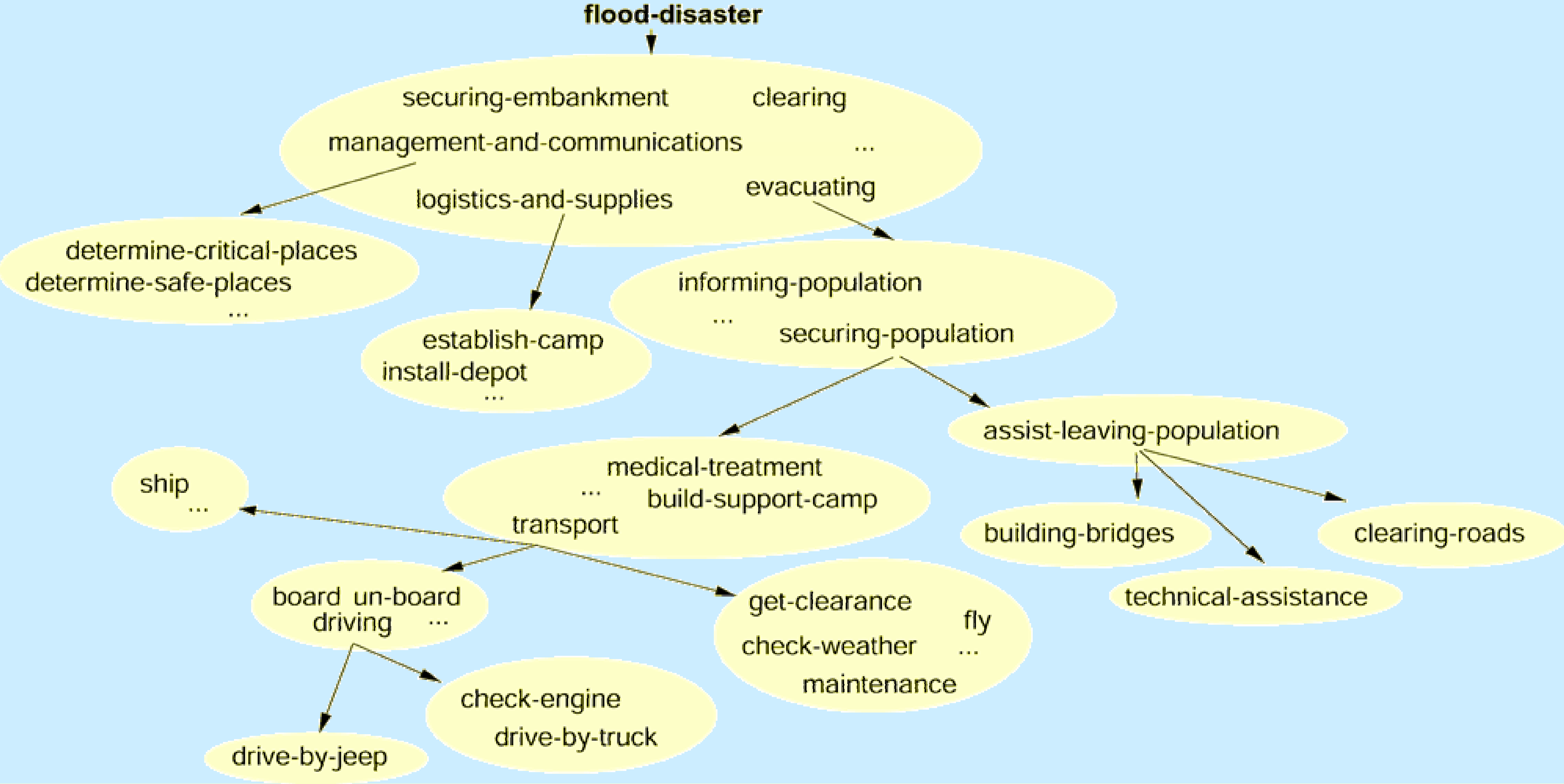
Tasks

- **logistics**
- **construction**
- **organisation**

180 groups, 1562 workers,

> 2 months mission

Example Tasks and Decompositions



Application Example: Mission Planning

- **Requirements:**
  - **representation of procedural knowledge**
  - **representation of declarative knowledge**
  - **representation of resources like time, material, tools, and manpower**
  - **hierarchies on tasks, state descriptions, and resources**

Hybrid Planning I

PANDA (Biundo & Schattenberg 2001)

combination of

- state- and operator-based abstraction
- declarative and procedural task descriptions

- preconditions and effects on any level of abstraction
- integration of POCL and decomposition planning
- well-founded combination of both techniques through the concept of decomposition axioms

Hybrid Planning II

- **decomposition axioms relate preconditions and effects of tasks across abstraction levels**

⇒ **decomposition of tasks**

⇒ **refinement of preconditions and effects**

$$\begin{aligned} \textbf{At (u:unit, l:loc)} \leftrightarrow [\ & \textbf{(V-At (u:vehicle, l:loc, r:road)} \wedge \textbf{. . .)} \vee \\ & \textbf{(A-At (u:aircraft, l:loc, h:height)} \wedge \textbf{. . .)} \vee \\ & \textbf{(C-At (c:container, l:loc)} \wedge \textbf{In (u:unit, c: ...)]} \end{aligned}$$

Hybrid Planning III

- **formal definition and proof of correctness of**
  - **methods**
  - **decomposition steps**

- **reasoning about resources on different levels of abstraction**

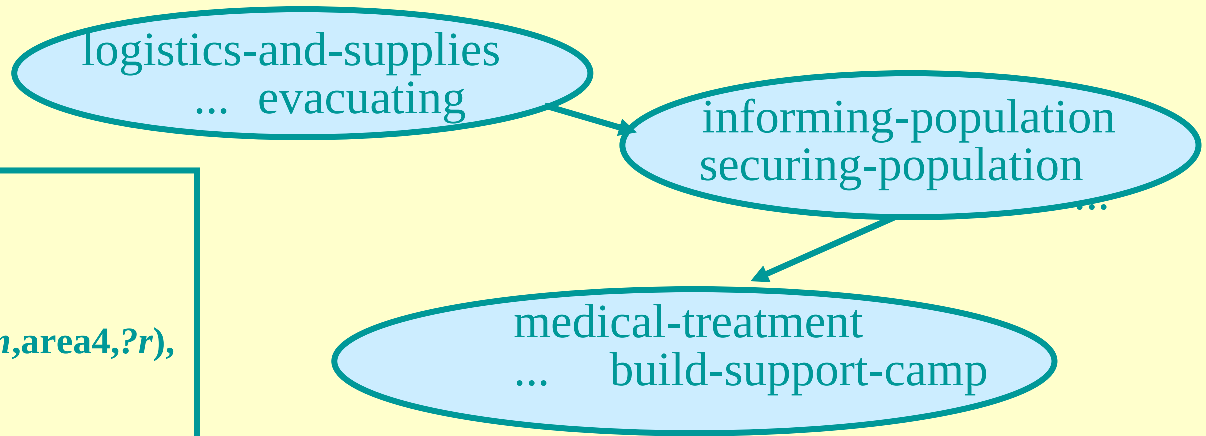
Energy-Checked (u:unit) $\leftrightarrow$ [(power-level (u:e-dev) > 300) $\vee$
(fuel-level (u:m-dev) > 100) $\vee$ ...]

Hybrid Planning IV

Planning Algorithm

- combining POCL planning and task decomposition
- extending the range of plan refinement operations
  - task decomposition
  - task insertion
  - causal link identification and establishment
  - variable constraints
    - (non-) codesignation
    - sort restriction

Hybrid Planning V



drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep1,area4),
+Status(area4,occup),
–Status(area4,cleared)...

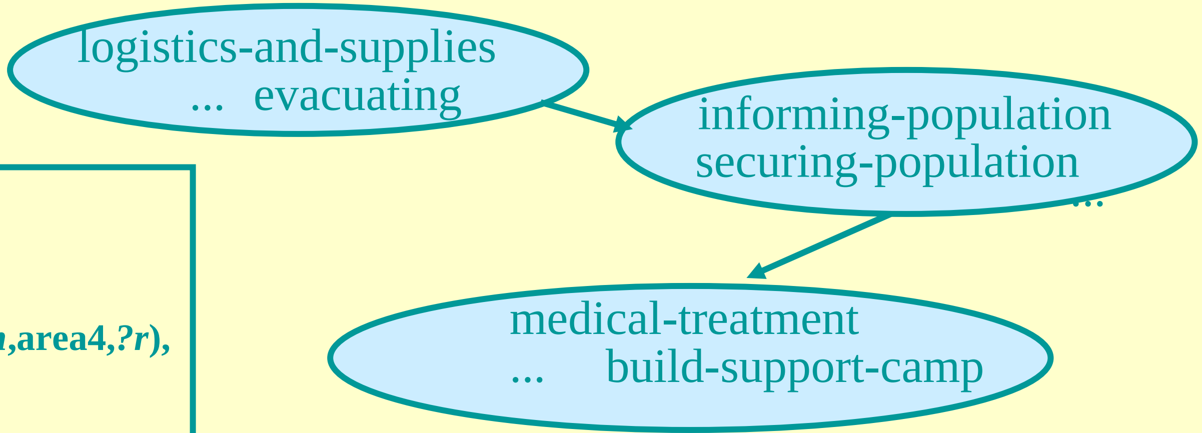
transport

(Passengers group1, Location area4,
Location camp2, Unit *?w*)
P: At(*?w*,area4),
At(group1,area4).
E: +At(*?w*,camp2),
+At(group1,camp2),
–At(*?w*, area4),
–At(group1, area4).

move

(Unit *?v*, Location *?from*,
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

Hybrid Planning V



drive

**(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).**

establish-camp

**(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep1,area4),
+Status(area4,occup),
–Status(area4,cleared)...**

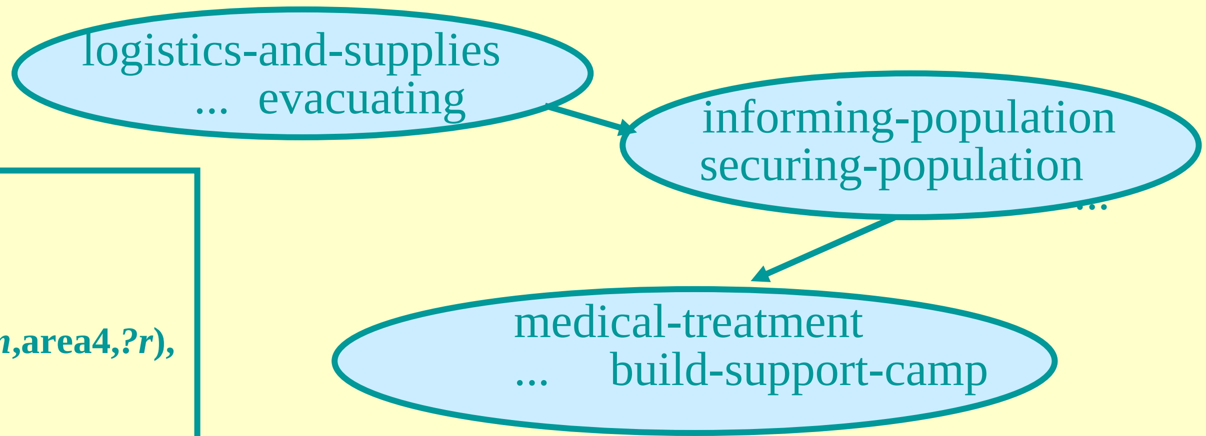
transport

**(Passengers group1, Location area4,
Location camp2, Unit *?v*)
P: At(*?v*,area4),
At(group1,area4).
E: +At(*?v*,camp2),
+At(group1,camp2),
–At(*?v*, area4),
–At(group1, area4).**

move

**(Unit *?v*, Location *?from*,
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).**

Hybrid Planning V



drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep1,area4),
+Status(area4,occup),
–Status(area4,cleared)...

transport

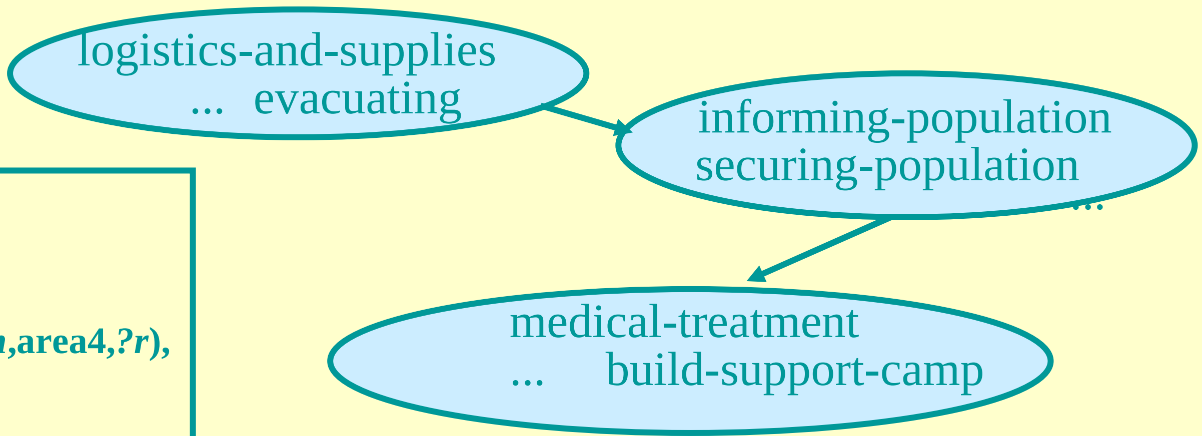
(Passengers group1, Location area4,
Location camp2, Unit *?w*)
P: At(*?w*,area4),
At(group1,area4).
E: +At(*?w*,camp2),
+At(group1,camp2),
–At(*?w*, area4),
–At(group1, area4).

move

classical

(Unit *?v*, Location ?
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

Hybrid Planning V



drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

move

classical

(Unit *?v*, Location ?
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

transport

(Passengers group1, Location area4,
Location camp2, Jeep jeep18)
P: At(jeep18,area4),
At(group1,area4).
E: +At(jeep18,camp2),
+At(group1,camp2),
–At(jeep18, area4),
–At(group1, area4).

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep18,area4),
+Status(area4,occup),
–Status(area4,cleared)...

Hybrid Planning V

logistics-and-supplies
… evacuating

informing-population
securing-population
…

medical-treatment
… build-support-camp

drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep18,area4),
+Status(area4,occup),
…tus(area4,cleared)...

transport

(Passengers group1, Location area4,
Location camp2, Unit *?w*)
P: At(*?w*,area4),
At(group1,area4).
E: +At(*?w*,camp2),
+At(group1,camp2),
–At(*?w*, area4),
–At(group1, area4).

move

(Unit *?v*, Locati…
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

classical

sort-hierarchy

Hybrid Planning V

logistics-and-supplies
… evacuating

informing-population
securing-population
…

medical-treatment
… build-support-camp

drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

transport

(Passengers group1, Location area4,
Location camp2, Vehicle *?u*)
P: At(*?u*,area4),
At(group1,area4).
E: +At(*?u*,camp2),
+At(group1,camp2),
–At(*?u*, area4),
–At(group1, area4).

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep18,area4),
+Status(area4,occup),
tus(area4,cleared)...

move

(Unit *?v*, Location
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

classical

sort-hierarchy

Hybrid Planning V

logistics-and-supplies
… evacuating

informing-population
securing-population
…

medical-treatment
… build-support-camp

drive

(Vehicle *?u*, Location *?from*,
Location area4, Road *?r*)
P: V-at(*?u*,*?from*,*?r*),
Reachable-by-land(*?from*,area4,*?r*),
Status(*?r*,ok).
E: +V-at(*?u*,area4,*?r*),
–V-at(*?u*,*?from*,*?r*).

decomposition axiom

establish-camp

(Location area4, Group thw26,
Jeep jeep18, Truck truck9, ...)
P: Status(area4,cleared), ...
E: +At(truck9,area4),
+At(jeep18,area4),
+Status(area4,occup),
tus(area4,cleared)...

sort-hierarchy

transport

(Passengers group1, Location area4,
Location camp2, Vehicle *?u*)
P: At(*?u*,area4),
At(group1,area4).
E: +At(*?u*,camp2),
+At(group1,camp2),
–At(*?u*, area4),
–At(group1, area4).

move

classical

(Unit *?v*, Location
Location area4)
P: At(*?v*,*?from*).
E: +At(*?v*,area4),
–At(*?v*,*?from*).

Hierarchy on Resources

PANDA extension (Schattenberg & Biundo 2002)

Integration of reasoning about resources

- **Different types of abstraction on resources**

  Aggregation   Subsumption

  Qualification   Approximation

- **Rationale**
  - **detect and resolve possible over-consumptions at any level of abstraction**
  - **prune the search space**
  - **guide the search towards "efficient" plans**

SAT-based Planning I

Kautz & Selman 1992

- **encoding the planning problem into a propositional formula**
  - **introduce time points**
  - **fix a plan length**
  - **generate ground instances of operator descriptions, goals, properties**

    $$\text{move}(\text{truck}, \text{loc}_1, \text{loc}_2, 1) \rightarrow [\, \text{At}(\text{truck}, \text{loc}_2, 2) \wedge \neg \text{At}(\text{truck}, \text{loc}_1, 2) \,]$$

    $$\text{move}(\text{truck}, \text{loc}_1, \text{loc}_2, 1) \rightarrow [\, \text{At}(\text{truck}, \text{loc}_1, 1) \,]$$

  - **encode ground atoms a propositional variables**

SAT-based Planning II

planning through satisfiability testing

- **construct a model**

  ⇨ **find an assignment of truth values to the variables such that the formula holds**

  ➡ **Davis-Putnam procedure**

  ➡ **stochastic procedures: GSAT, Walksat**

fail: **increase plan length**

success: **extract the plan from the model**

SAT-based Planning III

combining graph-based and SAT-based planning

- **construct a planning graph**
- **encode the planning graph in a propositional formula**
  - **time points are levels of the graph**
  - **facts of level i+1 imply the disjunction of level i actions that add them**
  - **$\neg$ o (t1, ..., tn, i) $\vee$ $\neg$ o' (t'1, ..., t'n, i) for mutex actions**
  - **prec imply actions etc., respectively**
- **construct a model of the formula**

Deductive Planning

Green 1969, Manna & Waldinger 1979

- first-order plan specifications
  - ⇒ existence formulas : $\forall x \, \exists y \; [\varphi(x) \rightarrow \psi(x, y)]$
- planning through theorem proving
  - ⇒ constructive proofs : $y \leftarrow p(x)$
- provably correct plans
- complex plans
  - ⇒ conditionals (case analysis)
  - ⇒ recursion (induction)

planning as programming

I didn’t mention ...

- **... planning and plan execution**
- **... integrating planning and scheduling**
- **... planning through constraint satisfaction**
- **... probabilistic planning**
- **... planning for information gathering**
- **... planner performance evaluation and comparison**
- **... ...**

Historical Aspects