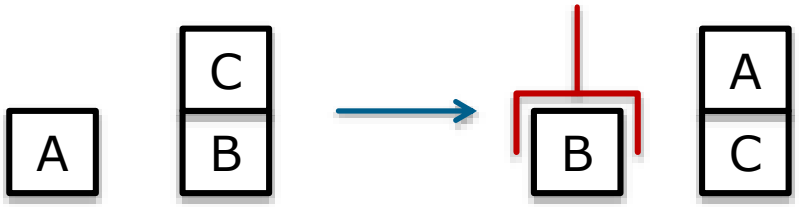
Planning Systems



Prof. Dr. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

Content

- Foundations of robot planning
- Forms of planning
- Planning via searching
- Cranfield-assembly-benchmark
- Summary of foundations

Forms of Planning in Robot Applications

- Action planning
  - Which actions are derived from tasks?
- Sequence planning
  - Temporal relationship between actions
- Resource planning
  - Which robots/devices execute actions?
- Time planning
  - When are actions started?
- Execution planning
  - Which parameters for action execution (trajectory- and grasp planning)?
- Scheduling: Resource and time planning

Planning in Robotics: Planning System

- Known initial state
- Desired goal state
- Generates action plan (sequence of actions), which stepwise transforms the system from initial to goal state
- Supervision of plan with help of sensors
- Re-planning if deviation detected

Planning in Robotics: Plan

- Action sequence or graph produced by planner
- Describes a set of state transitions in state-space which lead to a goal state
  - State transitions can be sequential, concurrent or coordinated-concurrent
- Can be divided into sub-plans
  - Plan segments: complete sub-plan
  - Plan skeletons: incomplete sub-plan
- Action/plan primitive: Atomic plan segment or state transition
- Linear plan: Comparable to linear decomposition of manipulated variable for minimization of controlled system‘s goal deviation

Planning Methods

- Planning: Finding a sequence of steps in order to arrive at a goal situation from an initial situation
- Chain of state transitions can be calculated, derived, searched for, or chosen
- Planning methods process exclusively symbol sets
- Senseless plans possible if knowledge base contains erroneous states or infeasible state transitions

Planning Methods

- Description of a planning problem
  - Initial state
  - Possible actions (operations), that can change the state
  - Goal state (or set of goal states)
- Easiest case: Plan is simple sequence (or acyclic graph) of actions
- General case: Any (known from programming languages) control structure possible

Edge- and Pre-conditions

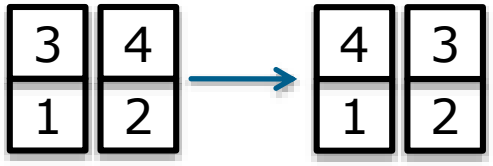
- Not every action possible in given situation
- Example grasping: Robot can grasp if…
  - Object is reachable
  - Object‘s mass movable by robot
  - Grasper in open position
  - Grasper at object‘s grasping position
  - Grasper appropriate for grasping object

Constraints in Assembly Planning

- Minimal processing time or in constant time span
- As few tool changes as possible
- Mechanical stability of partially assembled components
- Form of bonding (e.g. consider curing times when gluing)
- Material characteristics (e.g. deformability)
- Observability (e.g. camera)
- Precision of relative positioning of parts

Forms of Planning

- Linear: Strict temporal sequence of execution
- Non-linear: Complex temporal relations of actions (sequential, overlapping or parallel execution)

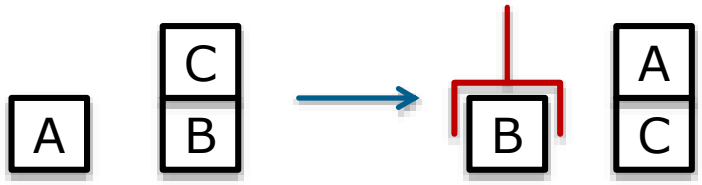


Placing on table not possible

- Monotonic: Every part is immediately placed in final position (no immediate placements)
- Not monotonic: With intermediate placements
- Central: Complete knowledge about all sub-processes
- Distributed: Multi-agent-systems
- Single-stage: Plan generates actions directly
- Multi-stage: Plan generates sub-goals

Compatibility of Sub-Goals

- Initial state: A and B on table, C on B
- Goal state: A on C, B in grasper
- Sub-goals:
  - A on C, B in grasper
  - First A on C → First sub-goal prevents second
  - First B in grasper → Second sub-goal prevents first



Fundamental Concept of Planning

- Z: Set of possible states
- Initial state $M_a \in Z$, goal state $M_e \in Z$
- Set O of actions α with precondition V_α
- State transformation $g_\alpha: Z \to Z$ describes state transition if V_α valid
- Searched: Sequence M_i of states with $M_1 = M_a$, $M_n = M_e$
  $\forall i = 1..n, \exists \alpha_i \in O$ with $M_{i-1} \succ V_{\alpha_i}$ and $g_{\alpha_i}(M_{i-1}) = M_i$
- Solution
  - Plan generation through Sequence of logical operations
  - Building of a tree with states as nodes and edges as actions
  - Usage of search methods for plan generation (depth-first search, breadth-first search, heuristic search)

Planning as Logical Mapping

- Solving the planning problem through derivation using a logical calculus if defined as follows: „Can the goal state be reached from the current state?“
- Used calculus: $1st$ order predicate logic or situation calculus v
- Derivation of reachable states through planning by inference
- Planning through monotonic logical derivation for more complex problems or reactive planning systems has only limited use
- Many phenomena of the real world and real environments in which agents act are not strictly monotonously derivable

Planning as Logical Mapping

- For most practical purposes non-monotonous logic has to be extended with exception rules
  - This makes it unnecessary to implement a planner for an autonomous robot as a pure „logical proof system“
- To be able to derive if a state is reachable from another, the states need to be formally describable
- Assuming a set of sensors or observers, which can indicate characteristics of individuals and their relations among each other. Then the set of characteristics and relations defines the current state of the process.

Planning as Logical Mapping

- $\text{holds}(x_j, y)$: In process state y characteristic x_j holds
- Terms can be constructed using logical connectives and quantifiers
- Terms constructible, which make universally valid statements
- New characteristics of individuals and their relations are derivable, even if they are not directly measurable using sensors
- Example: If x_j and x_k hold, then x_l holds.

Planning as Logical Mapping

- For process description (interaction, action, behavior) the situation calculus introduces a result operator (result). When in process state y an operation o or action u for state transition is detected, then a new situation $\text{result}(u, y)$ arises.
- This allows for establishing the axiom, saying that for every state y and action u the state $\text{result}(u, y)$ is reachable. This state is also reachable from every state y_s, from which y is reachable.

$$\forall y: \text{reachable}(y, y)$$
$$\forall y, u: \text{reachable}(\text{result}(u, y), y)$$
$$\forall y, y_s, u: \text{reachable}(\text{result}(u, y), y_s) \Leftrightarrow \text{reachable}(y, y_s)$$

Planning as Logical Mapping

- This axiom allows for deriving the reachability of a state from another.
- Depending on the chosen inference mechanism, the realization of planning through derivation can correspond to a search (Prolog depth search)
- Description of proof problem:
  „Demonstrate that there exists at least one arbitrary state y_z that can be reached from another arbitrary state y_s and y_s is defined by the characteristic x_s and y_z by x_z "

$$\forall y_s : \text{holds}(x_s, y_s) \Rightarrow \exists y_z : \text{holds}(x_z, y_z) \wedge \text{reachable}(y_z, y_s)$$

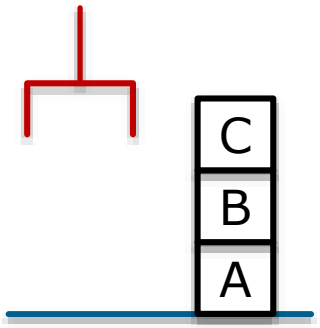
Planning as Logical Mapping: Example Situation Calculus

- Proof system: Derives if a plan exists that transfers the initial state (a) to the goal state (b)
- The goal state is defined by block A

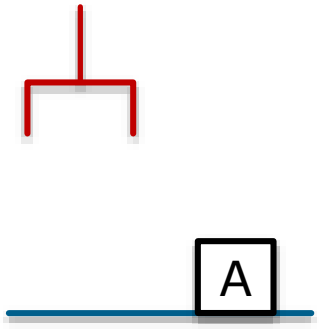
$$\mathrm{holds}\big(\mathrm{clear}(A, y_z)\big)$$

- The initial state y is defined by

$$holds(on(C,B), y_s) \wedge holds(on(B,A), y_s) \wedge holds(clear(C), y_s)$$



(a) Initial state



(b) Goal state

Planning as Logical Mapping: Example Situation Calculus

- Rule for stacking motion

$$\forall y: \text{holds}(\text{clear}(top), y) \wedge \text{holds}(\text{on}(top, bot1), y) \wedge \text{holds}(\text{clear}(bot2), y)$$
$$\Rightarrow \text{holds}\big(\text{on}(top, bot2), \text{result}(\text{puton}(top, bot2), y)\big)$$
$$\wedge \text{holds}\big(\text{clear}(bot1), \text{result}(\text{puton}(top, bot2), y)\big)$$

- Condition to be proven

$$\forall y_s: \text{holds}(\text{on}(C, B), y_s) \wedge \text{holds}(\text{on}(B, A), y_s) \wedge \text{holds}(\text{clear}(C), y_s)$$
$$\Rightarrow \exists y_z: \text{holds}(\text{clear}(A), y_z) \wedge \text{reachable}(y_z, y_s)$$

- Result of application of derivation rules

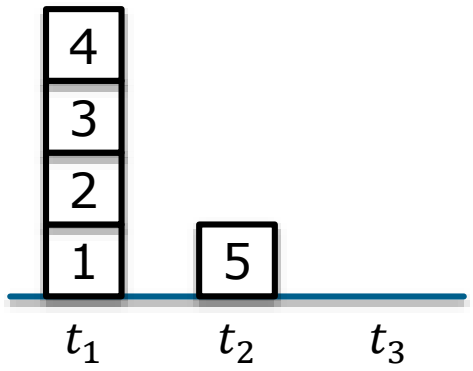
$$\forall y_s: \text{holds}(\text{on}(C, B), y_s) \wedge \text{holds}(\text{on}(B, A), y_s) \wedge \text{holds}(\text{clear}(C), y_s)$$
$$\Rightarrow \exists y_z: \text{holds}(\text{clear}(A), y_z)$$
$$\wedge \text{reachable}\left(y_z, \text{result}\big(\text{puton}(B, table), \text{result}(\text{puton}(C, table), y_s)\big)\right)$$

Example: Planning via Searching

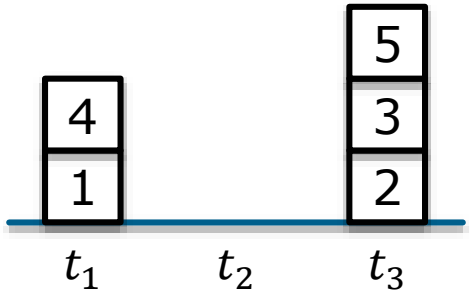
Possible plan

(1) Block 4 on block 5
(2) Block 3 on block 4
(3) Block 2 on t_3
(4) Block 3 on block 2
(5) Block 4 on block 1
(6) Block 5 on block 3

Initial state



Goal state



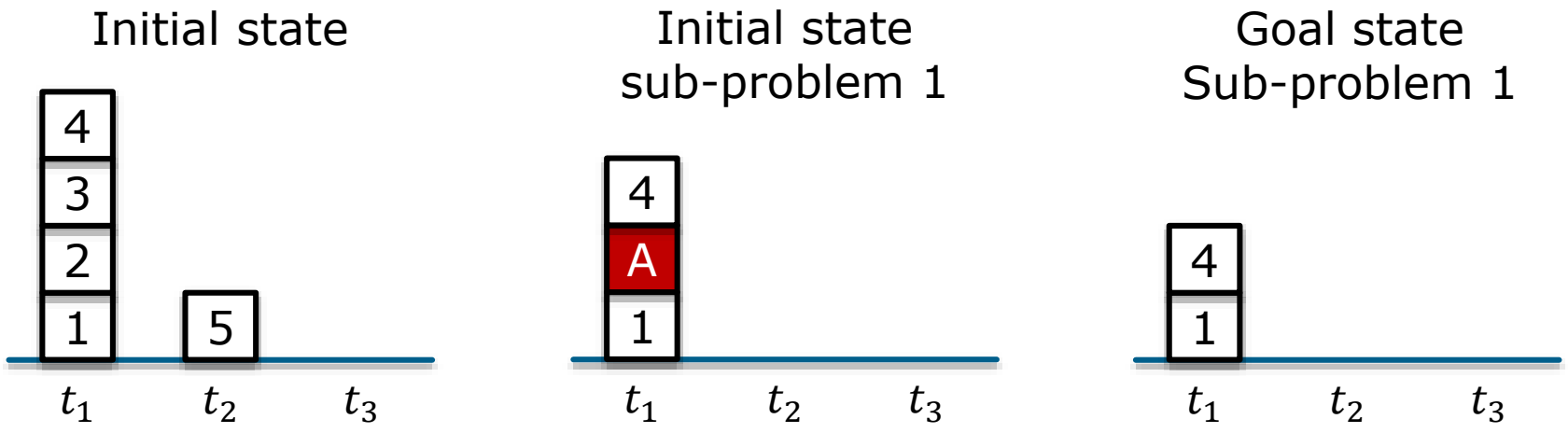
Example: Planning with Sub-Goals

- Sub-goal 1: $\mathrm{on}(1, t_x)$ and $\mathrm{on}(4,1)$
- Plan for sub-goal 1

(1) Block 4 on table

(2) Block A on table

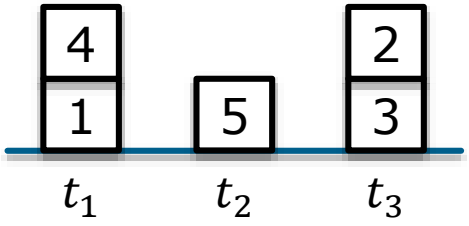
(3) Block 4 on block 1



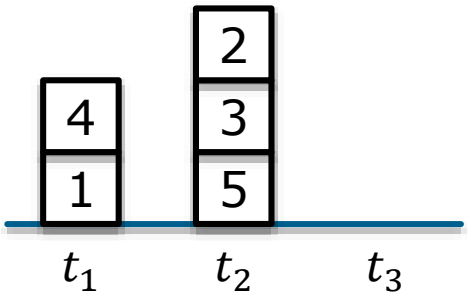
Example: Planning with Sub-Goals

- Sub-goal 2: $\mathrm{on}(2, t_y)$, $\mathrm{on}(3,2)$ and $\mathrm{on}(5,3)$
- Two possible initial states

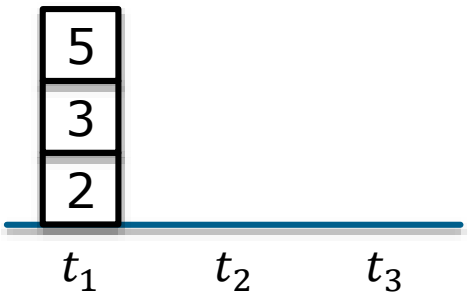
Initial state 1
sub-problem 2

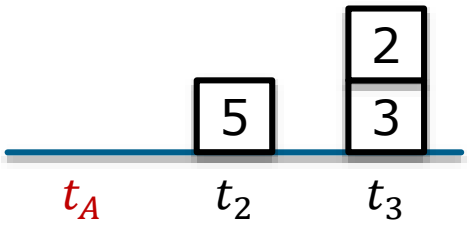


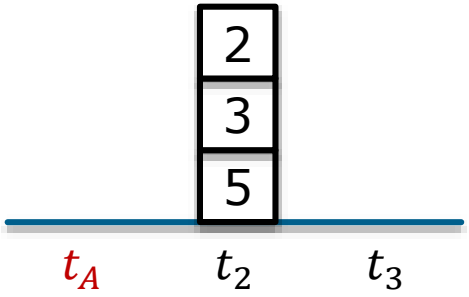
Initial state
sub-problem 1



Goal state
sub-problem 2





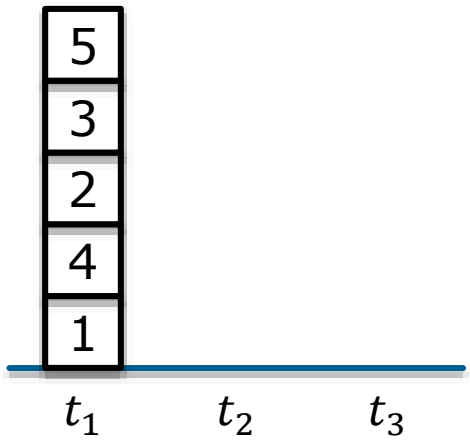


Abstract initial states

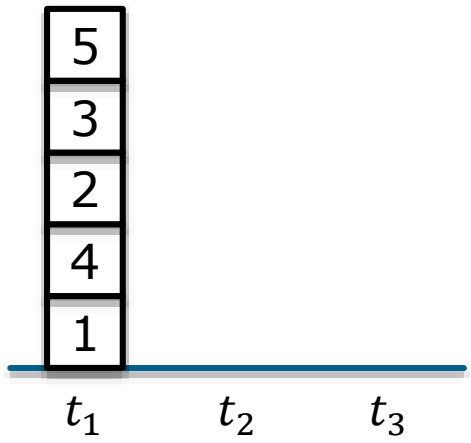
Example: Planning with Sub-Goals

- Plan for sub-goal 2
  (1) Block 2 on table
  (2) Block 3 on block 2
  (3) Block 5 on block 3
- Three possible solutions

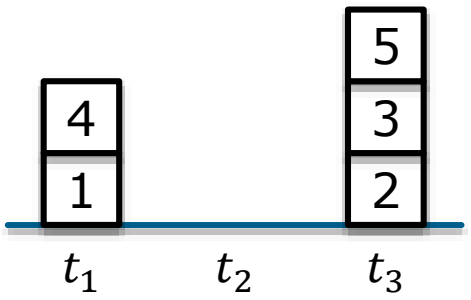
Solution 1
sub-problem 2



Solution 2
sub-problem 2



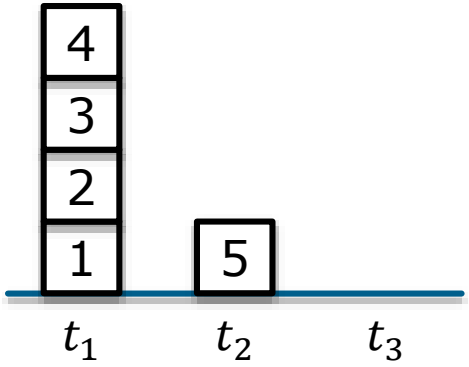
Solution 3
sub-problem 2



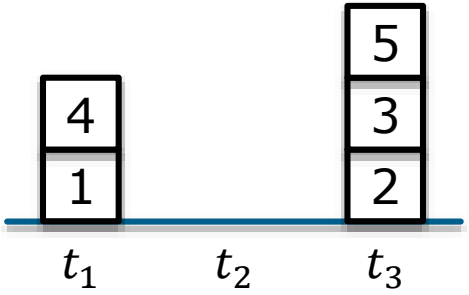
Example: Optimal Overall Solution

(1) Block 4 on t_3
(2) Block 3 on block 5
(3) Block 2 on block 3
(4) Block 4 on block 1
(5) Block 2 on t_3
(6) Block 3 on block 2
(7) Block 5 on block 3

Initial state

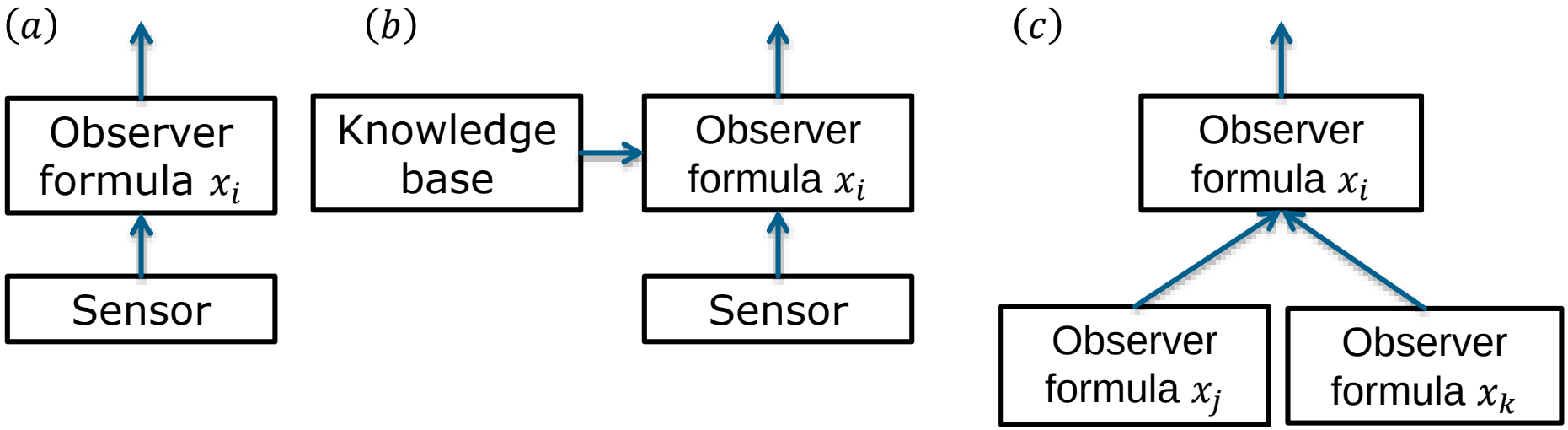


Goal state



Planning via Searching: STRIPS

- Logic formula x_i makes statements about characteristics/relations of process y
- Process state $x(y)$ as conjunction of formulae x_i
- $x(y) = \bigcup x_i$ with x_i holds in state y
- Based on information from sensors (a), stored knowledge (b) or application of calculus (c)



Planning via Searching : STRIPS

- States are manipulated via operators o or plans consisting of operators p
- An operator is a quadruple $o = (u, cond, add, sub)$
  - Description u: Semantic meaning of modification
  - Pre-condition $cond$: Logic formula, examines if operator is applicable to state
  - Add set add: Additional formulae of the new state compared to the original state
  - Sub set sub: Missing formulae of the new state compared to the original state

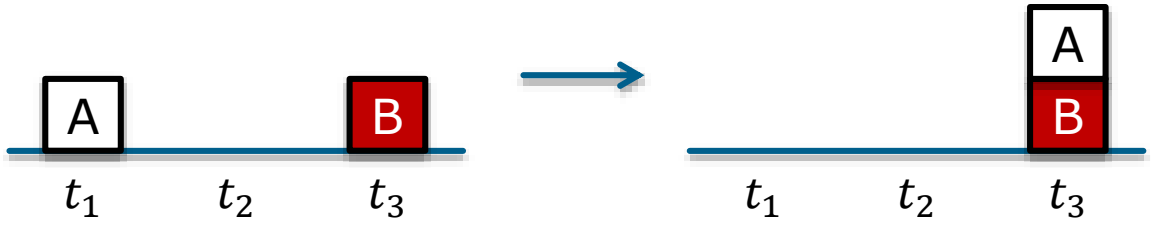
STRIPS: Representation of Block World

- Current state

$$x(y) = \{\text{clear}(A), \text{clear}(B), \text{red}(B)\}$$

- Operator-quadruple

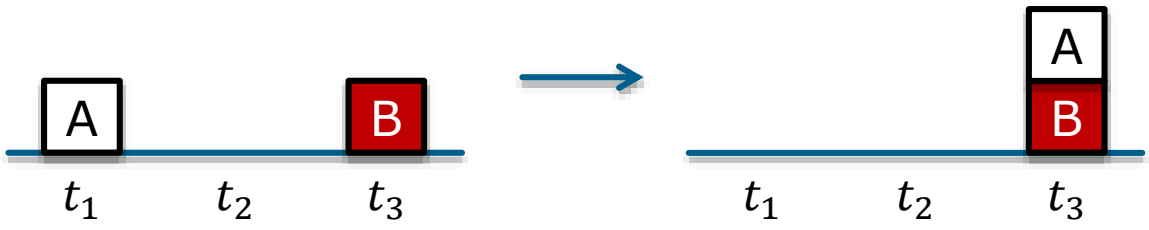
$$(\text{puton}(A, B), \{\text{clear}(A), \text{clear}(B)\}, \{\text{clear}(A), \text{on}(A, B)\}\{\text{clear}(B)\})$$



STRIPS: Representation of Block World

- Application of operator
  - $\text{puton}(A, B)$ can be applied, because $\{\text{clear}(A), \text{clear}(B)\}$ is valid
  - $\text{clear}(A)$ still holds
  - Additionally $\text{on}(A, B)$ holds
  - $\text{clear}(B)$ no longer valid
  - No statement regarding $\text{red}(B)$ is made
- Result of operator application

$$x(y) = \{\text{clear}(A), \text{red}(B), \text{on}(A, B)\}$$



STRIPS: Characteristics

- In the add-set, all formulae which hold in the new state need to be included
  - Example: If $\mathrm{clear}(A)$ were missing, $\mathrm{puton}(A, B)$ could lead to a state in which $\mathrm{clear}(A)$ is not valid
- In the sub-set, all formulae which do not hold in the new state anymore need to be included.

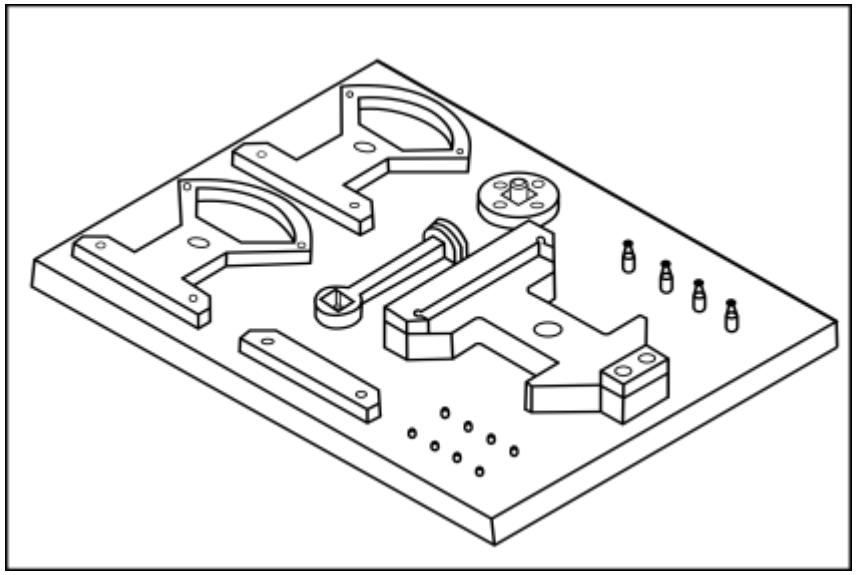
Planning Systems

- ASTRIPS (hierarchical)
- HACKER (single-stage)
- NOAH (hierarchical)
- ATLAS
- SHARP
- TWAIN (multi-stage)
- …

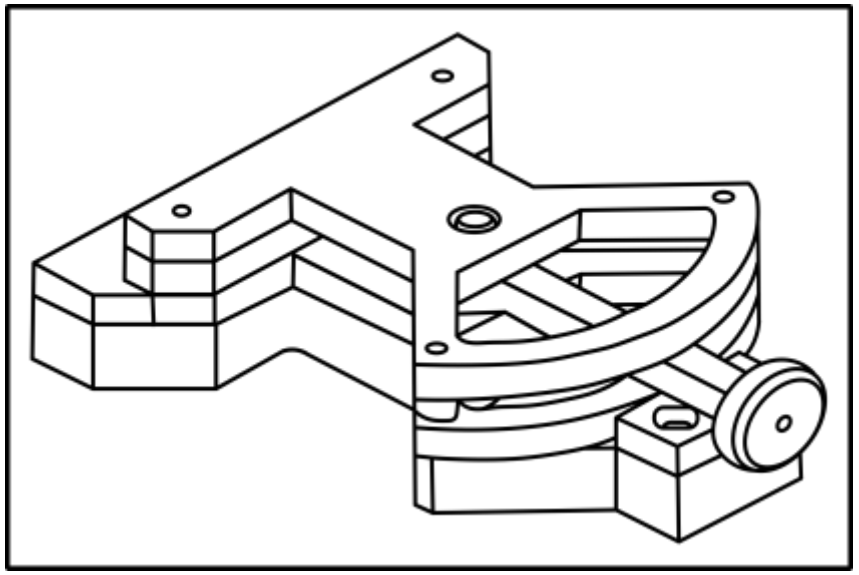
Example: Planning Phases for Simple Assembly

- World modelling
- Task specification
- Task analysis
- Building priority graph
- Planning of execution details
- Execution

Example: Cranfield-Assembly-Benchmark

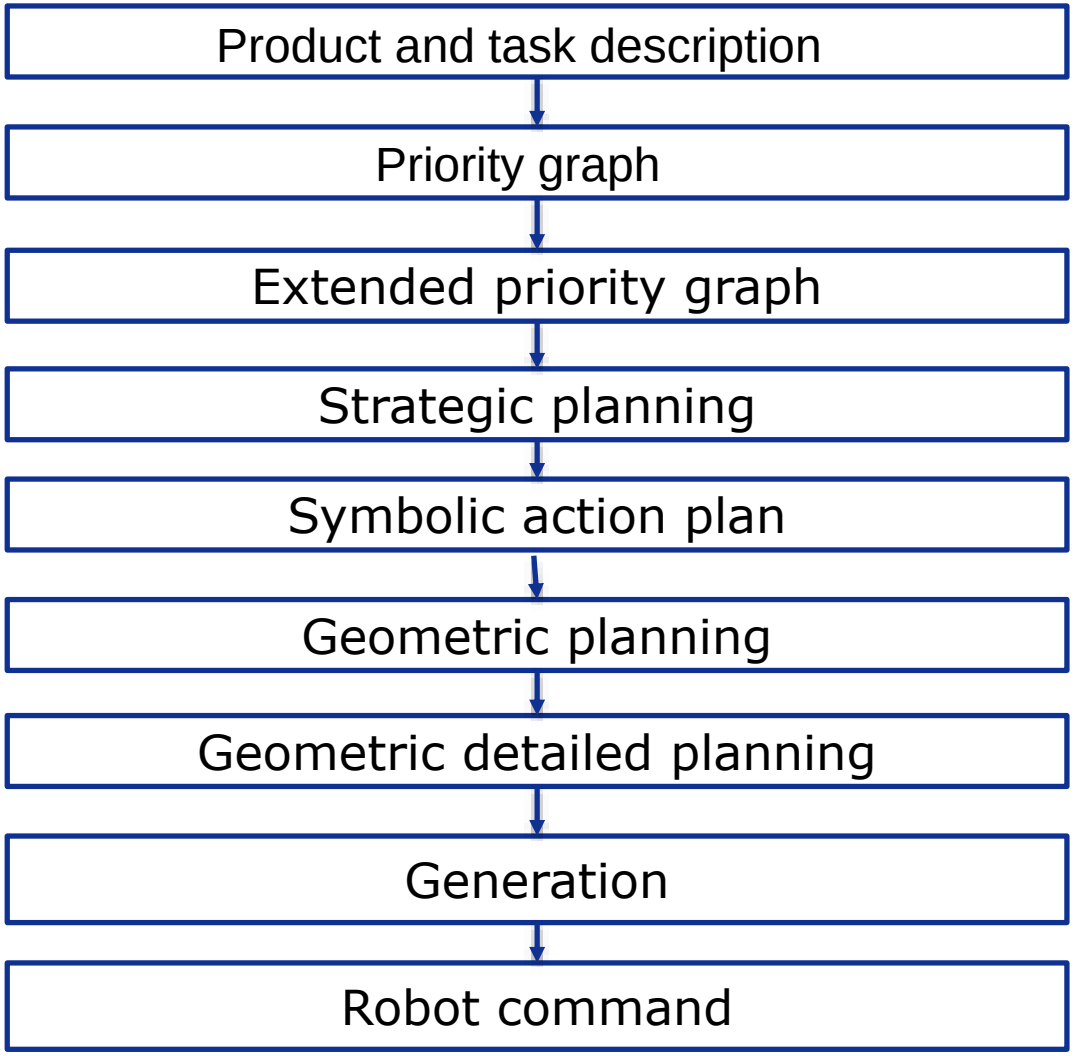


Initial configuration



End configuration

Example: Planning & Supervision of Assembly

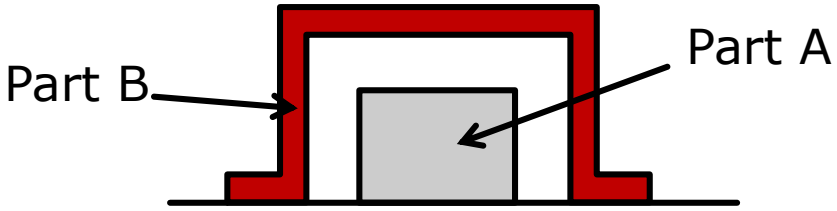


Task Specification

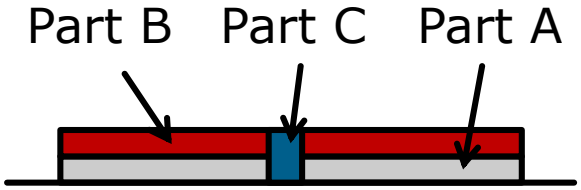
- Modelling of initial and end configurations of assembly
- Configuration
  - Geometry (3D)
  - Physical characteristics (Center-of-mass, material)
  - Cartesian coordinates of item
- Graphical editor for specification …
  - … of the item's position
  - … of the grasper configuration

Criteria for Task Analysis

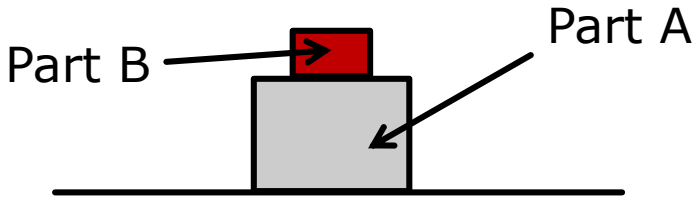
- No intersection



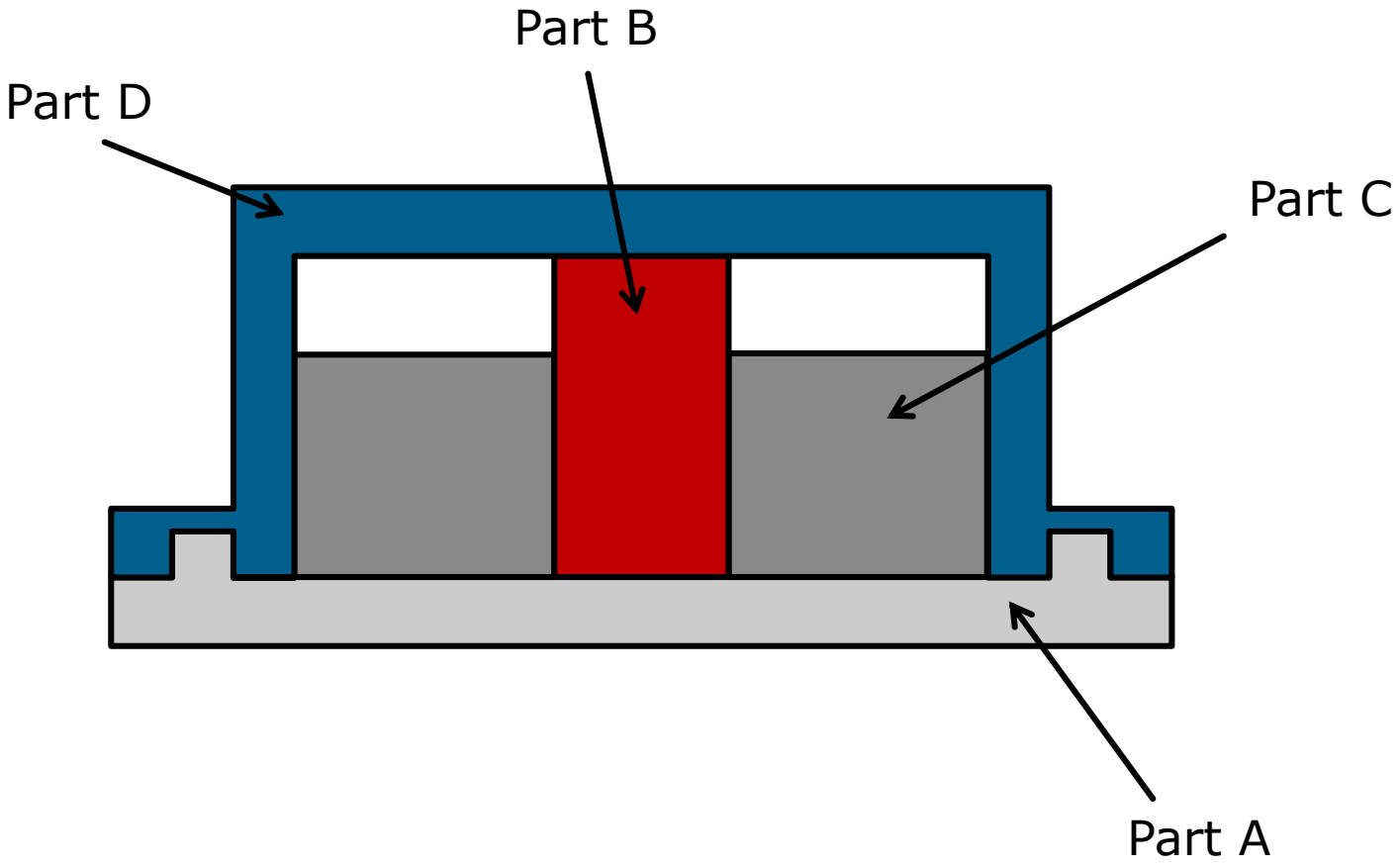
- No side-effects



- Stability



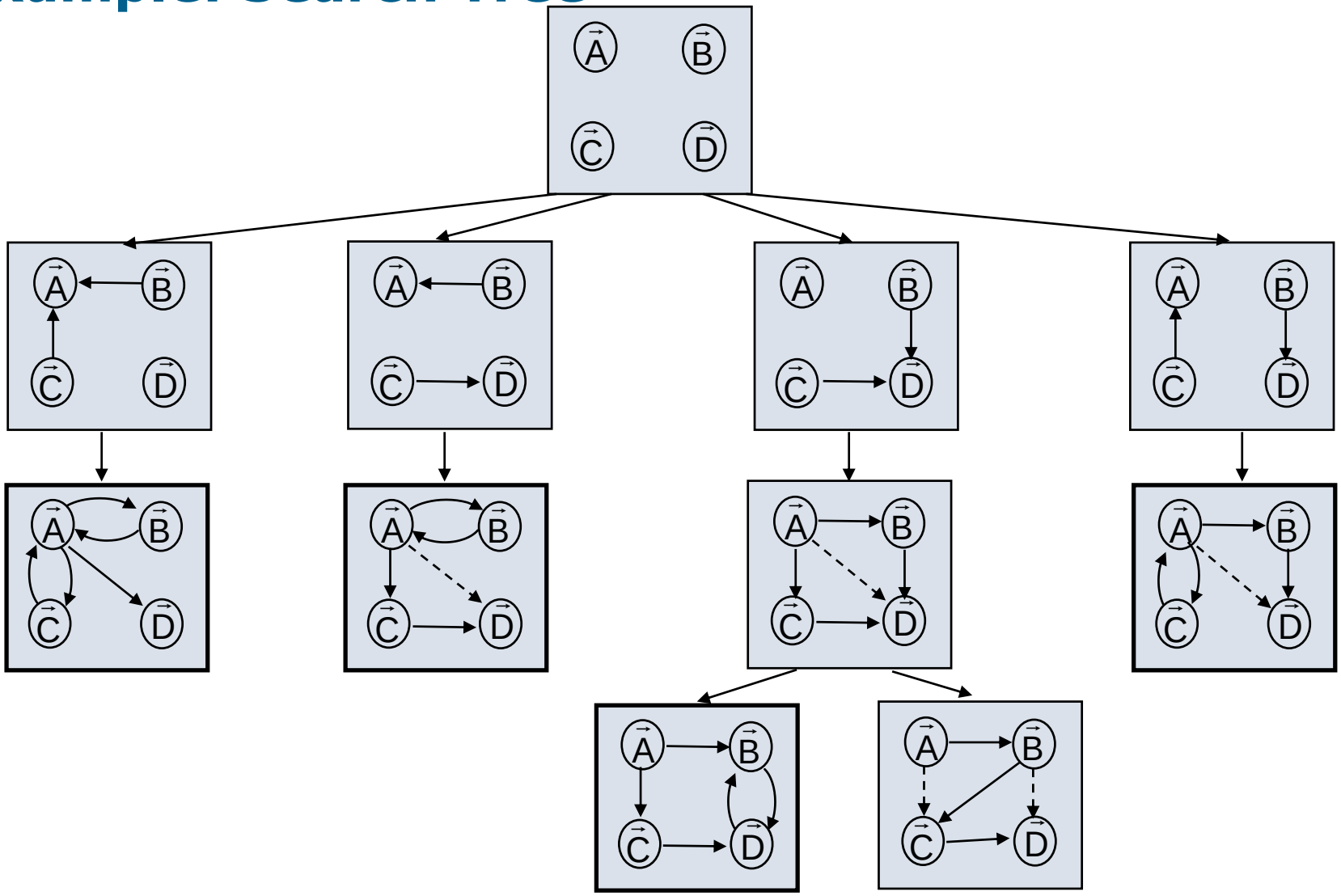
Example: Arrangement



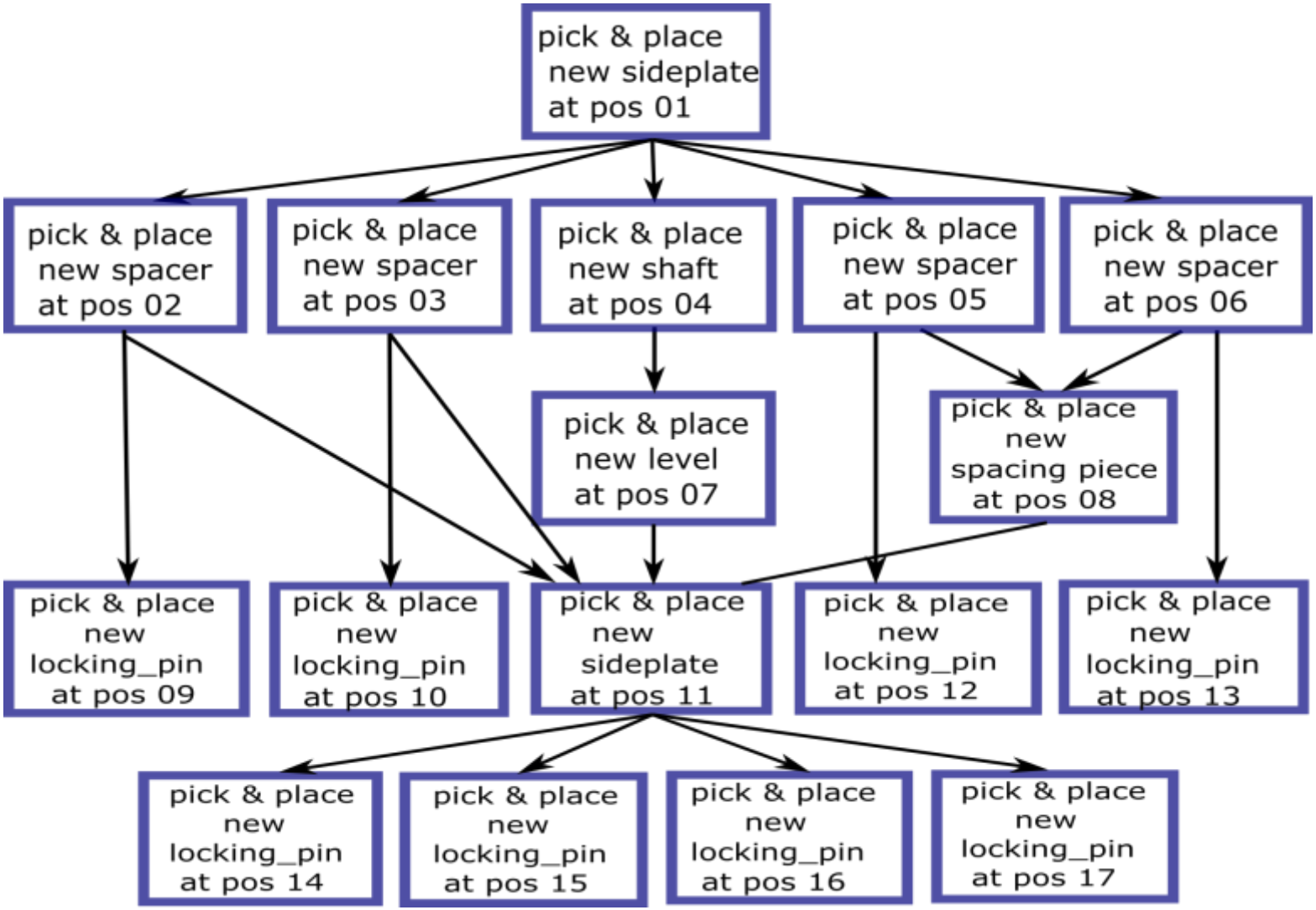
Example: Priority Conditions

Constrain source	Resulting priority condition
Intersection rule	$\text{OR}\left(\text{AND}\left(\pi(\vec{B},\vec{A}),\pi(\vec{C},\vec{A})\right)\right),$ $\text{AND}\left(\pi(\vec{B},\vec{A}),\pi(\vec{C},\vec{C})\right),$ $\text{AND}\left(\pi(\vec{B},\vec{D}),\pi(\vec{C},\vec{D})\right),$ $\text{AND}\left(\pi(\vec{B},\vec{D}),\pi(\vec{C},\vec{A})\right)$
Stability rule	$\text{AND}\left(\pi(\vec{A},\vec{B}),\pi(\vec{A},\vec{C}),\pi(\vec{A},\vec{D})\right)$
Side-effect rule	$\text{OR}\left(\pi(\vec{B},\vec{C}),\pi(\vec{D},\vec{B})\right)$

Example: Search Tree



Priority Tree of Cranfield-Benchmarks



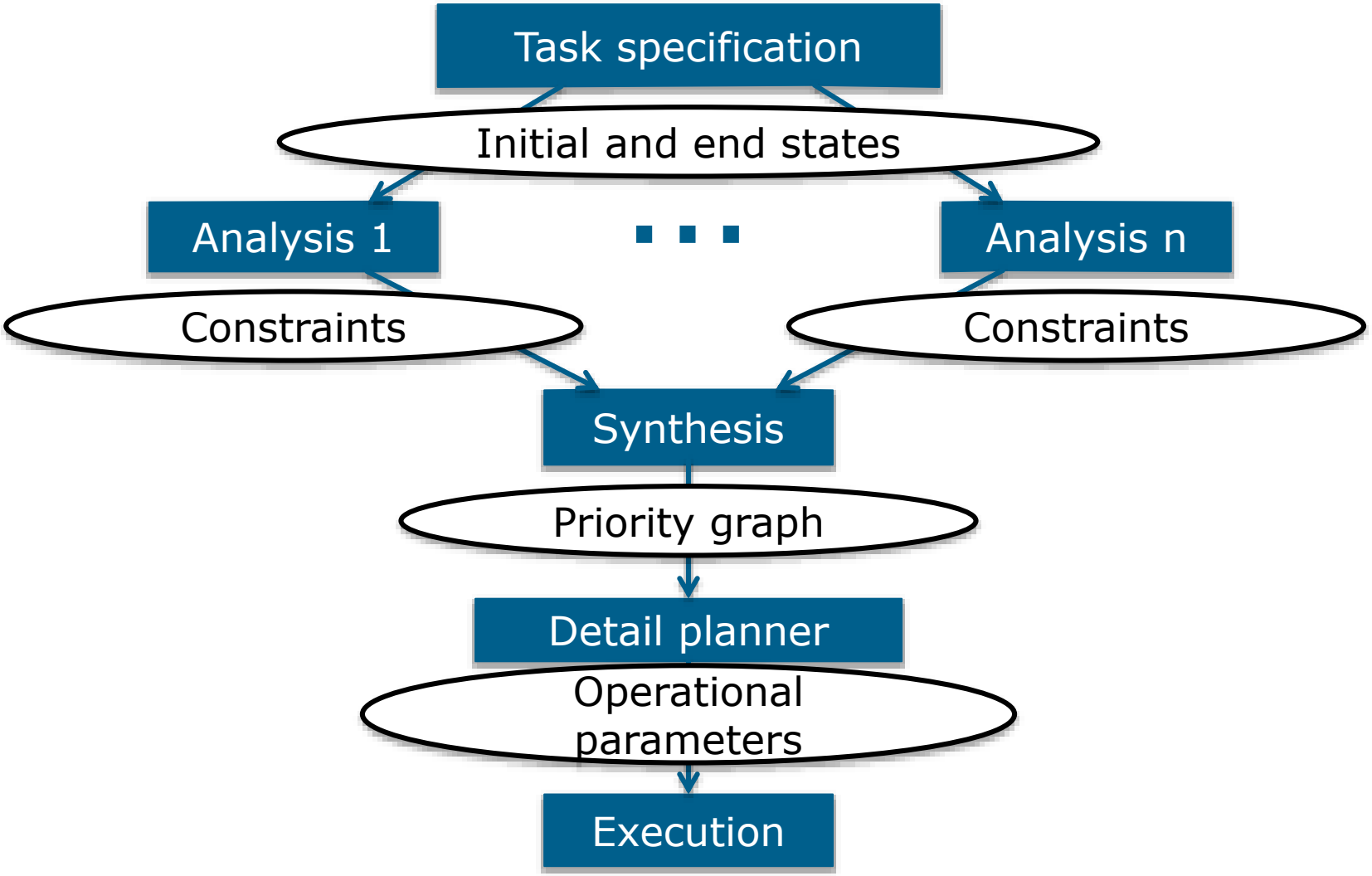
Detailed Planning

- Choice of robot
- Layout planning
  - Finding a robot pose
  - Criteria: Reachability, accuracy and efficiency
- Motion planning
  - Collision-free motion for grasping and releasing of items
  - Collision-free minute motion to induce targeted contact of two items
  - Collision-free rough manipulator motion for transporting items

Synthesis of Priority Graph

- Fusion of rule set arising from task analysis
- Choice of „best“ priority graph
- Elimination of redundant edges
- Cycle detection

Overall System



Next Lecture...

Robot architectures

- Capabilities of a robotic system
- Hierarchically function-oriented architecture
- Distributed function-oriented architecture
- Hierarchically behavior-oriented architecture
- Distributed behavior-oriented architecture