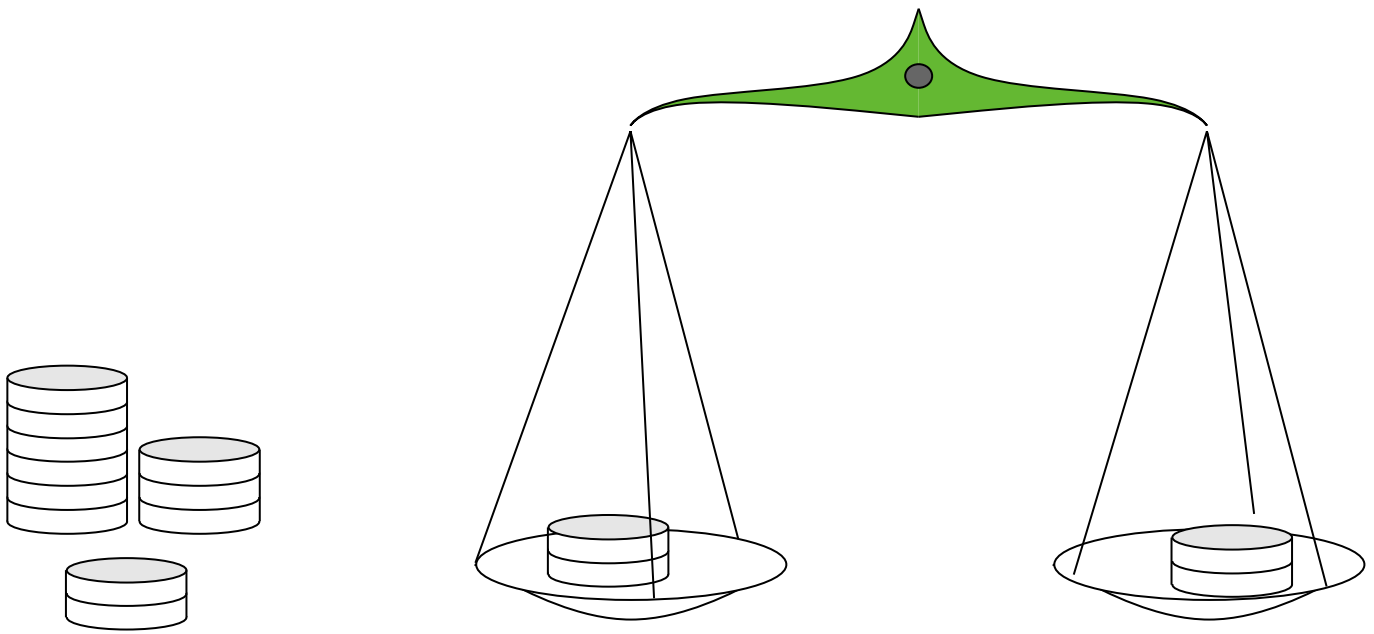
Chapter S:II

II. Search Space Representation

- Systematic Search
- Encoding of Problems
- State-Space Representation
- Problem-Reduction Representation
- Choosing a Representation

Problem-Reduction Representation

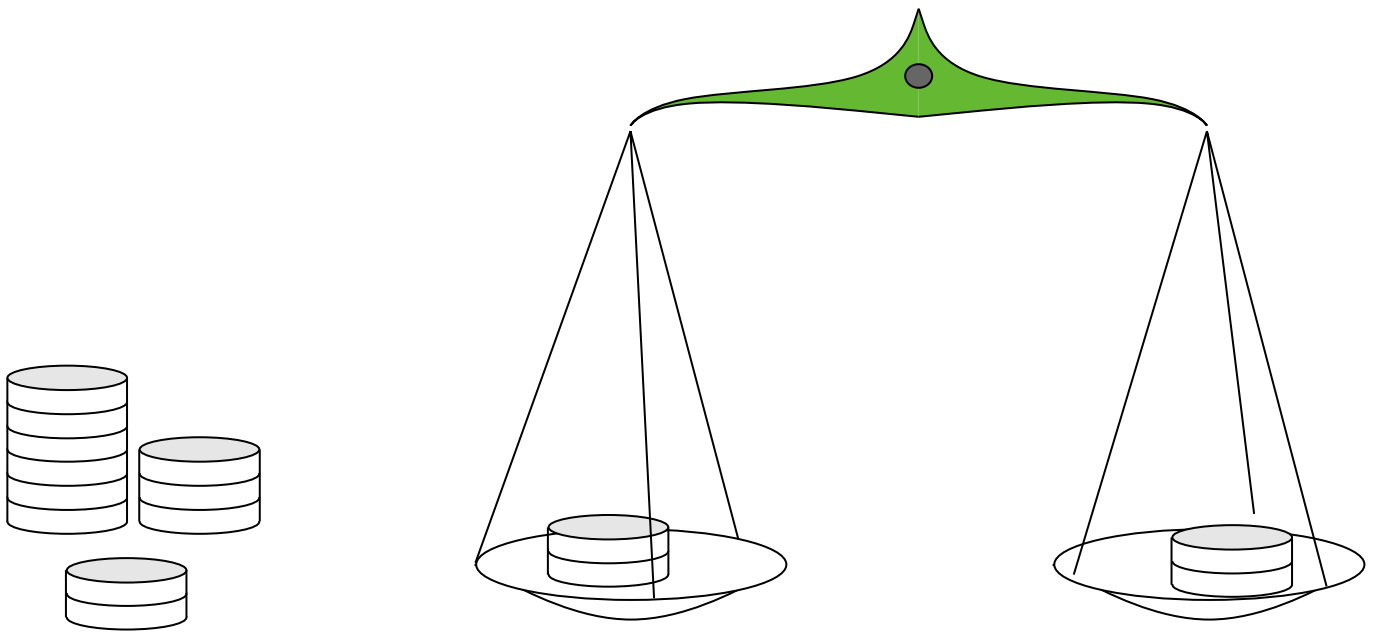
The Counterfeit Coin Problem



- 12 coins are given, one of which is known to be heavier or lighter.
- 3 weighing tests are allowed to find the counterfeit coin.

Problem-Reduction Representation

The Counterfeit Coin Problem



- Sought: a weighing strategy
  What to weigh first, second, etc.?
  How to process different weighing outcomes?
- Solution: exhaustive search
- The role of heuristics: focus on the most promising weighing strategy.

Remarks:

- Each rest problem (specified as state) can be described by knowledge already acquired by the preceding tests. A coin belongs to one of the following four categories: not-heavy, not-light, not-suspect, unknown (= none of the first three categories).
- A weighing action is the selection of a number of coins (for each pan the same number) and to test the selection.
- After each test the category information of each coin is updated. E.g., if some coin is currently assigned to “not-light” and from the weighing outcome follows “not-heavy”, its new category becomes “not-suspect”.
- Each weighing action leads to one of three possible outcomes all of which must be dealt with. The most suitable weighing strategy depends on the objective to optimized:
  1. If the maximum number of weighing action is to be minimized, treat the most difficult outcome next.
  2. If the expected number of weighing action is to be minimized, treat the most likely outcome next.

Problem-Reduction Representation

The Different Link Types

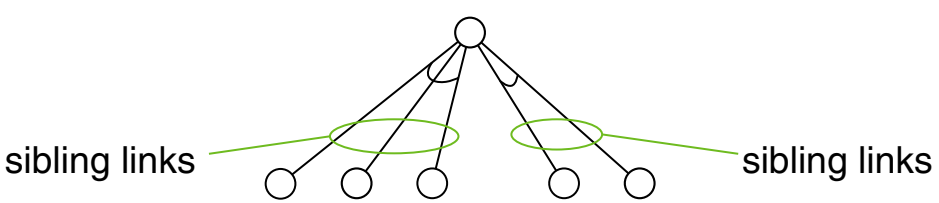
The application of an operator may lead to different kinds of subproblems. A graph that captures such a search space structure has two kinds of links:

1. AND links.
   Lead to independent subproblems all of which, however, must be solved in order to solve the problem associated with the parent node.
2. OR links.
   Lead to alternative subproblems one of which has to be solved in order to solve the problem associated with the parent node.

A graph of this type is called problem-reduction graph or AND-OR graph.

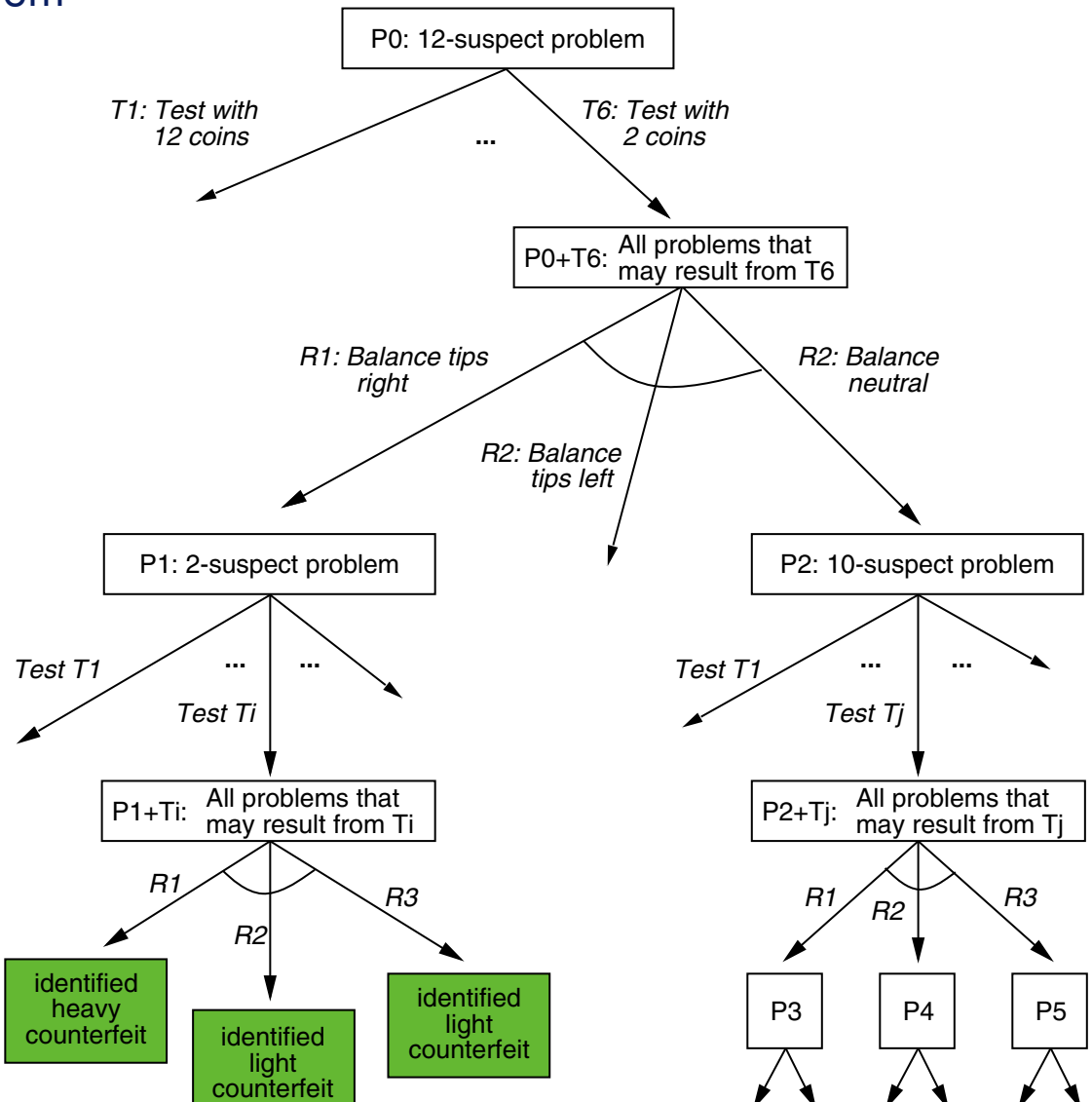
Remarks:

- As in the state-space graph: OR links encode applied operators (problem transformation).
- As in the state-space graph: The sequence of OR links along a path that starts at the root node define a solution base.
- Q. Does a path from the root node to a leaf node represent a solution to the problem?
- AND links model a problem decomposition. They are employed to decompose a problem according to possible outcomes (as in the counterfeit problem) or to decompose a complex problem into less complex subproblems. [Towers of Hanoi]
- Several possibilities may exist to decompose a given rest problem (at a parent node) into subproblems. The AND links of those subproblems that belong together (sibling links) are hence marked as such.



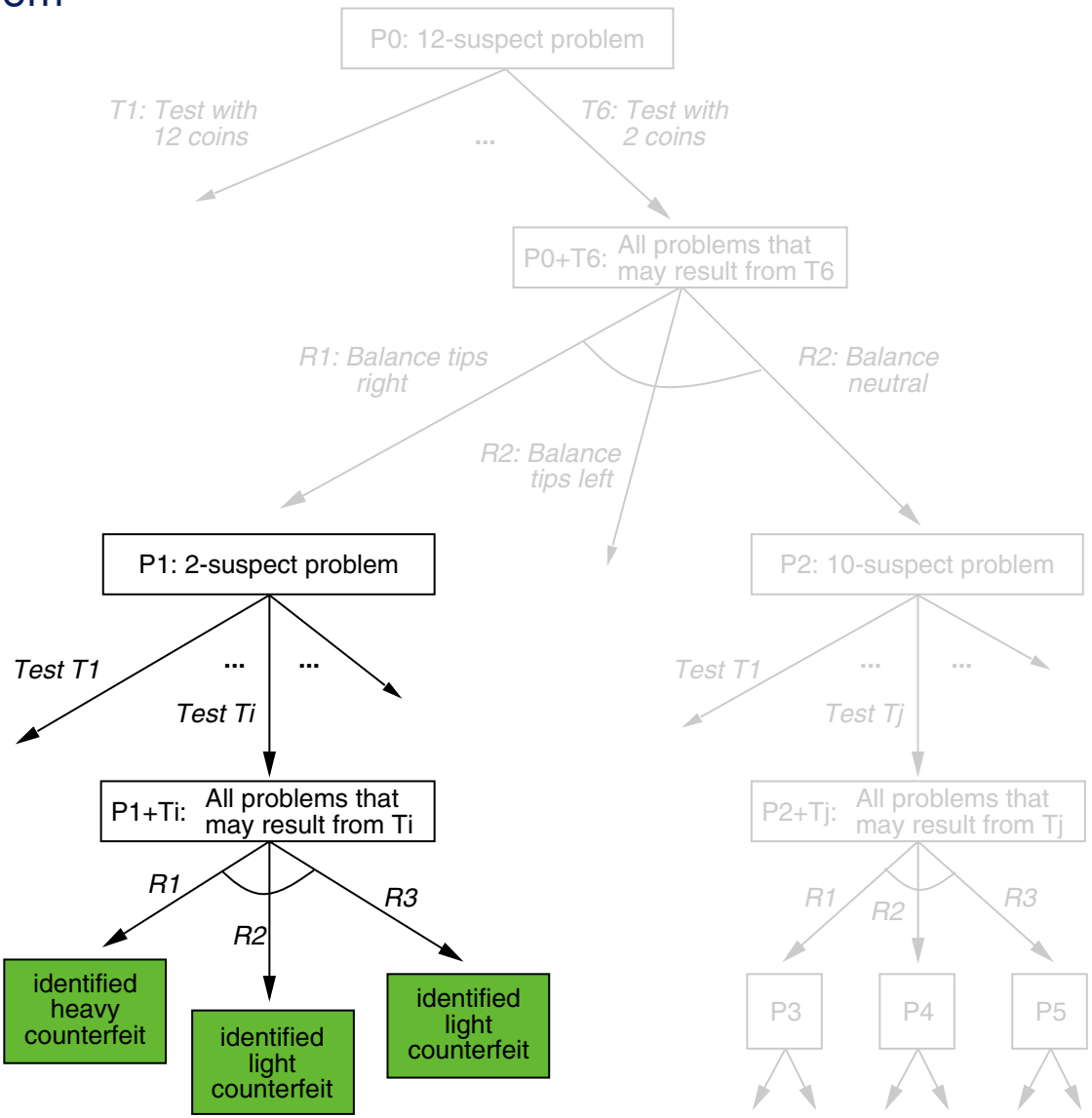
Problem-Reduction Representation

Counterfeit Problem



Problem-Reduction Representation

Counterfeit Problem



Remarks:

- In the 2-suspect problem, for instance, we have two coins for which the balance tips one way or another, along with ten remaining coins from which we know that they are honest.
- The 2-suspect problem (for instance) may occur several times in a weighing strategy.
- The 2-suspect problem (for instance) can be solved independently from all other weighing problems.
- The solution of the 2-suspect problem (for instance) can be determined, saved, and reused.

Problem-Reduction Representation

The Different Node types

The application of an operator may lead to different kinds of subproblems. A canonical graph that captures such a search space structure has two kinds of nodes:

1. AND nodes.
   Nodes with outgoing AND links only; more precisely: AND links that belong together, sibling AND links.

   Examples: decisions of an opponent (Keyword: game tree search), random processes in the nature, (artificial) experiments, results of external computations, all kinds of events controlled by a third party

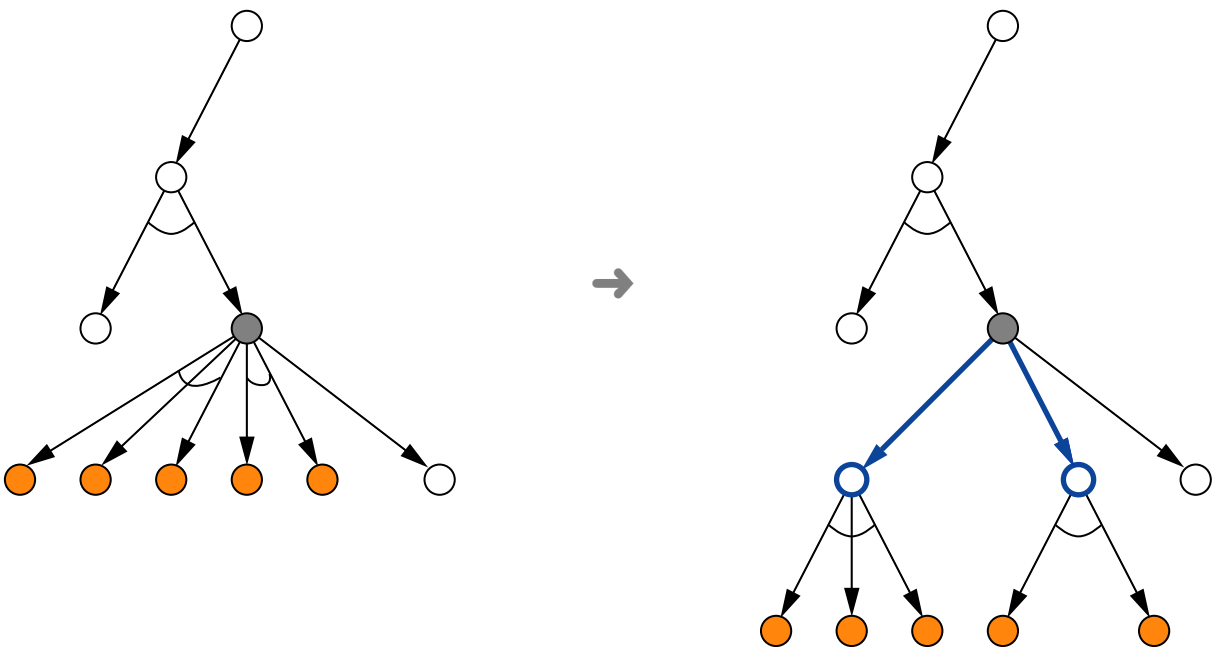
2. OR nodes.
   Nodes with outgoing OR links only.

   Examples: the own decisions, the actual strategy

Problem-Reduction Representation

Canonical Representation of AND-OR Graphs

If nodes are incident with different link types or with multiple groups of AND links, a canonical representation can always be generated:



In the following we will consider only AND-OR graphs that have a canonical representation.

Remarks:

- The AND-OR graph of the counterfeit problem has a canonical representation by nature.
- In the AND-OR graph of the counterfeit problem the child nodes of an OR node are AND nodes and vice versa. This alternation of node types is typical for action-reaction problems, e.g. represented in the form of a game tree. [S:VII Game Playing Introduction].
- The canonical representation of AND-OR graphs does not require alternating node types. In particular, the root node may not be of OR type.

Problem-Reduction Representation

Search Building Blocks [State-Space]

Definition 5 (Problem-Reduction Graph, AND-OR Graph)

Consider the set of all problems that can be generated by applying either the operators or a problem decomposition to the database.

One obtains the *problem-reduction graph* or *AND-OR graph* by

1. connecting each parent node with its generated successors using directed links,
2. labeling those links that belong to an operator with the applied operator,
3. comprising those links that belong to a problem decomposition as sibling links, and
4. introducing additional intermediate nodes and OR-links for each set of sibling AND-links if a node would have more than one set of outgoing sibling links or outgoing AND-links together with outgoing OR-links.

Problem-Reduction Representation

Solution Graph [State-Space]

Typically, a solution in an AND-OR graph is not a path but a more complex subgraph, called *solution graph*.

Definition 6 (Solution Graph in a Problem-Reduction Graph)

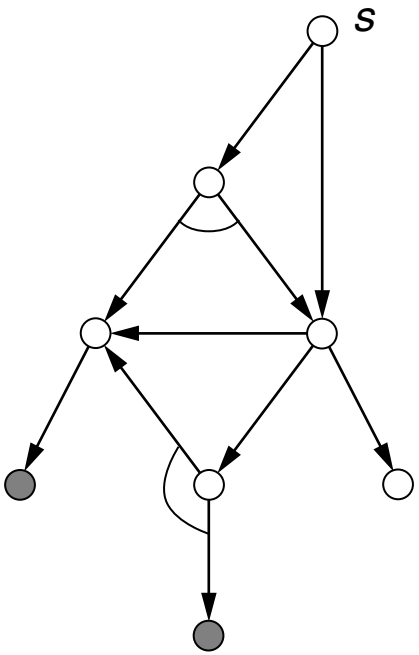
Let G be an acyclic AND-OR graph, and let n be a node in G. A finite subgraph H of G is called a *solution graph for* n *in* G iff ($\leftrightarrow$) the following conditions hold:

1. H contains the node n.
2. If H contains an inner OR node, then H contains exactly one link to a successor node in G and this successor node.
3. If H contains an inner AND node, then H contains all links to its successor nodes in G and all these successor nodes.
4. The leaf nodes in H are goal nodes (nodes with solved or trivial rest problems) in G.
5. H is minimal: it contains no additional nodes and edges.

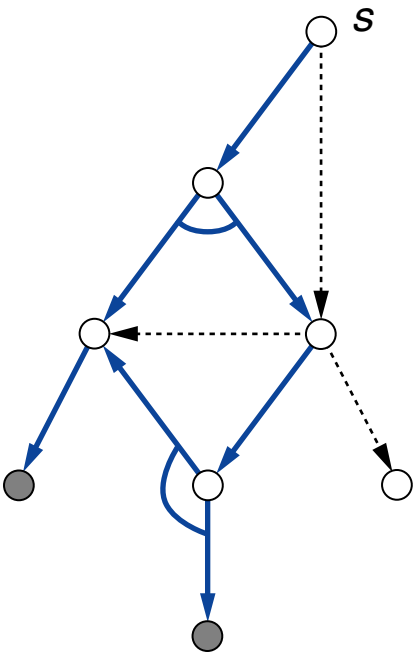
Problem-Reduction Representation

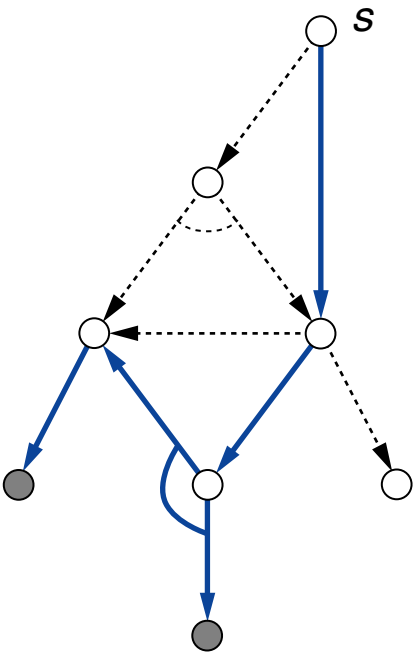
Solution Graph (continued)

AND-OR graph example:



Two possible solution graphs:





Remarks:

- Usually, we are interested in finding a solution pgraphs H for the root node s in G.
- If H is a solution graph for a node n in G and if n' is some node in H, then the subgraph of H rooted at n', H', is a solution graph for a node n' in G. This subgraph is called the *solution graph in H induced by n'*.
- Each solution graph defines a decomposition hierarchy of the problem associated with n.
- The instantiation of a solution graph is based on local decisions. E.g., a solution graph for some AND node is given by combining solution graphs of its successors.
- A solution graph is compact in the sense that it does not contain multiple occurrences of identical rest problems.
- A legitimate solution in a problem-reduction graph must be finite. This property is implicitly fulfilled since the problem-reduction graph itself is created by a search algorithm within a finite amount of time.

Remarks (continued):

- The conditions for the determination of a solution graph in Definition 6 are not constructional but declarative. A constructional procedure for bottom-up solution graph synthesis could look like follows:

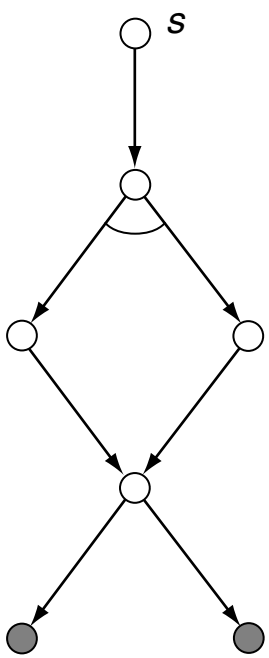
  1. If $n \in \Gamma$, i.e., n is a node that represents a solved rest problem, then $H_n := \langle \{n\}, \emptyset \rangle$ is a solution graph for n in G.
  2. If n is an OR node with successor n' in G and $H_{n'} = \langle V', E' \rangle$ is a solution graph for n' in G, then $H_n := \langle V' \cup \{n\}, E' \cup \{(n, n')\} \rangle$ is a solution graph for n in G.
  3. If n is an AND node with successors $n'_1, \ldots, n'_k$ in G and $H_{n'_i} = \langle V'_i, E'_i \rangle$ is a solution graph for n'_i in $G, i = 1, \ldots, k$, then $H_n := \langle V'_1 \cup \ldots \cup V'_k \cup \{n\}, E'_1 \cup \ldots \cup E'_k \cup \{(n, n'_1), \ldots, (n, n'_k)\} \rangle$ is a solution graph for n in G.

  Why is the above process problematic?

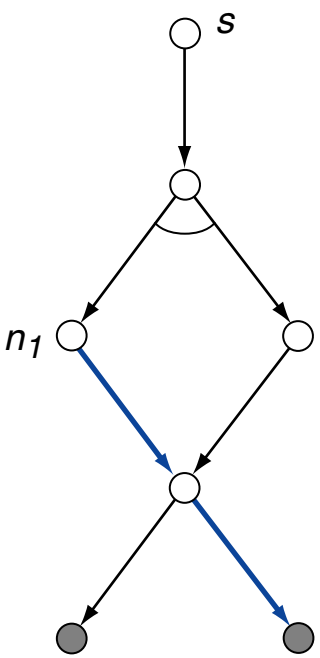
Problem-Reduction Representation

Solution Graph (continued)

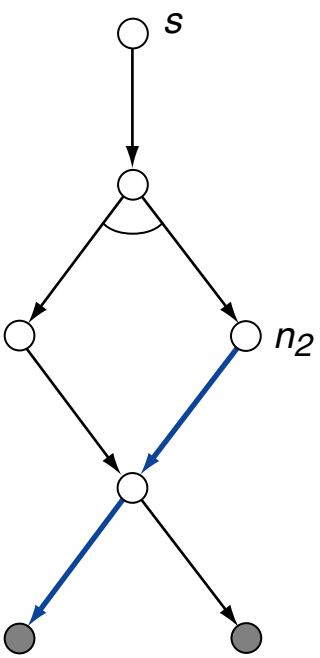
Synthesized solution graphs are not necessarily minimum:



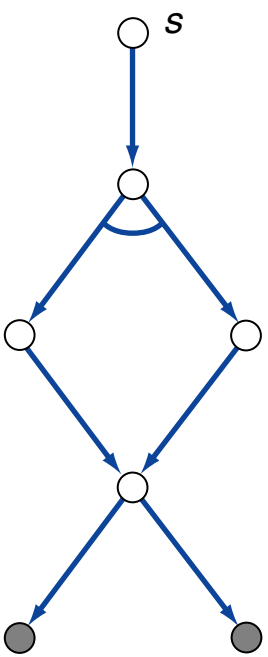
AND-OR graph



Solution graph for n_1



Solution graph for n_2



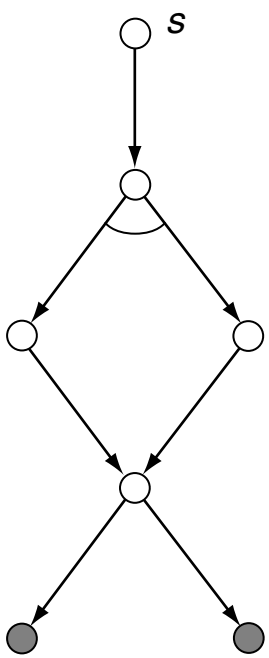
Solution graph for s

➜ The inductive formulation given subsequently as “solved-labeling procedure” will address this issue by node labeling.

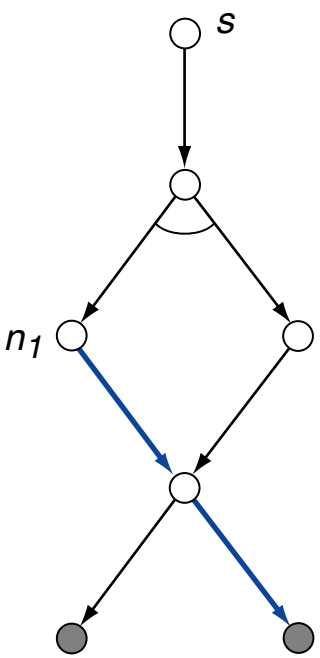
Problem-Reduction Representation

Solution Graph (continued)

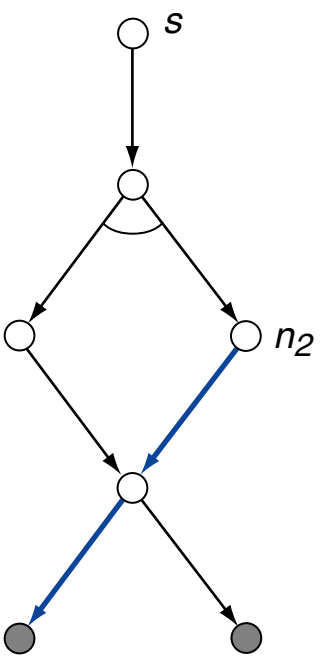
Synthesized solution graphs are not necessarily minimum:



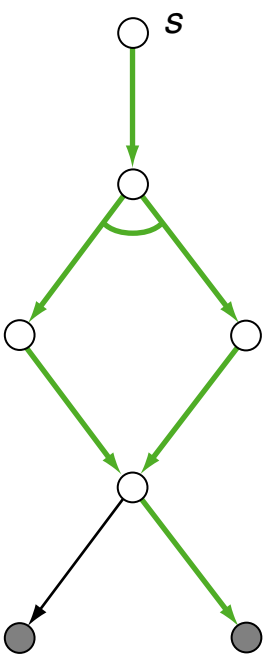
AND-OR graph



Solution graph for n_1



Solution graph for n_2



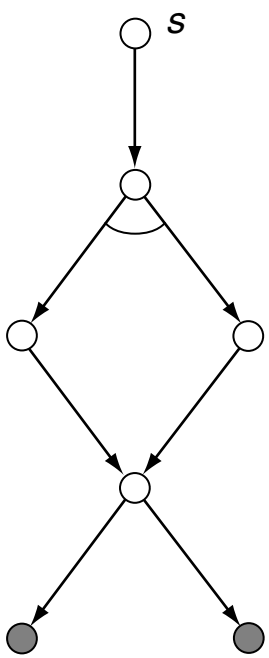
Minimum solution graph for s

➜ The inductive formulation given subsequently as “solved-labeling procedure” will address this issue by node labeling.

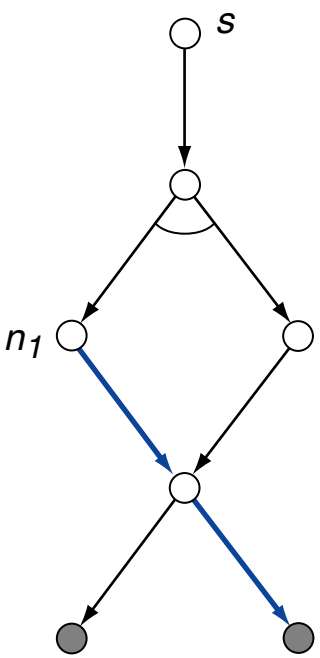
Problem-Reduction Representation

Solution Graph (continued)

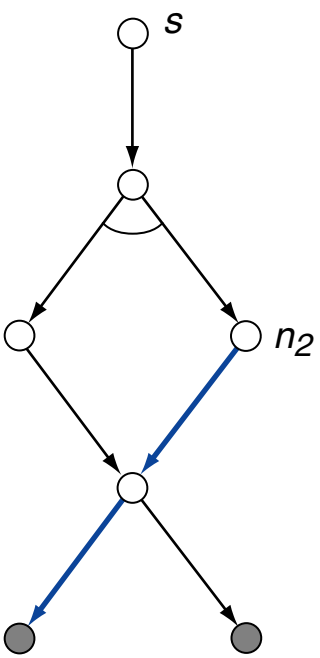
Synthesized solution graphs are not necessarily minimum:



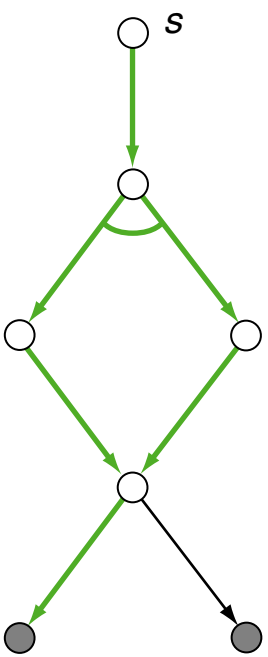
AND-OR graph



Solution graph for n_1



Solution graph for n_2



Minimum solution graph for s

➜ The inductive formulation given subsequently as “solved-labeling procedure” will address this issue by node labeling.

Problem-Reduction Representation

Solution Graph (continued)

The question whether or not an AND-OR graph G contains a solution graph H can be answered by a recursive application (depth-first search) of the following labeling rules, starting with the root node s.

Definition 7 (Solved-Labeling Procedure)

Let G be an acyclic AND-OR graph. A problem associated with a node n in G is labeled as "solved", if one of the following conditions is fulfilled.

1. n is a leaf node (terminal node) that represents a solved rest problem.
2. n is a nonterminal OR node, and at least one of its OR links points to a node labeled as "solved".
3. n is a nonterminal AND node, and all of its AND links point to nodes labeled as "solved".

Remarks:

- Usually, solved-labeling is not applied to an underlying search space graph G. Instead, it is applied to the finite, explored subgraph of G as a termination test: If the root node s is labeled “solved”, the explored part of G contains a solution graph.

Problem-Reduction Representation

Algorithm: solved-labeling

Input: G. A finite, acyclic AND-OR graph.
n. A node in G.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.

Output: The symbol *True* if a solution exists, *False* otherwise.

Problem-Reduction Representation

Algorithm: solved-labeling (without labeling)

Input: G. A finite, acyclic AND-OR graph.
n. A node in G.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.

Output: The symbol *True* if a solution exists, *False* otherwise.

`solved-labeling`$(G, n,$ *successors*$, \star)$

```
1.    IF |successors(n)| = 0 THEN
        IF ⋆(n)
        THEN RETURN(True);
        ELSE RETURN(False);
```

Problem-Reduction Representation

Algorithm: solved-labeling (without labeling)

Input: G. A finite, acyclic AND-OR graph.
n. A node in G.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.

Output: The symbol *True* if a solution exists, *False* otherwise.

`solved-labeling`$(G, n,$ *successors*$, \star)$

```
1.   IF |successors(n)| = 0 THEN
       IF ⋆(n)
       THEN RETURN(True);
       ELSE RETURN(False);
2.   FOREACH n' IN successors(n) DO
       IF solved-labeling(G, n', successors, ⋆)
       THEN
         IF OR_node(n) THEN RETURN(True); // One success is enough.
       ELSE
         IF AND_node(n) THEN RETURN(False); // One fail is one too much.
     ENDDO
```

Problem-Reduction Representation

Algorithm: solved-labeling (without labeling)

Input: G. A finite, acyclic AND-OR graph.
n. A node in G.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.

Output: The symbol *True* if a solution exists, *False* otherwise.

`solved-labeling`$(G, n,$ *successors*$, \star)$

```
1.  IF |successors(n)| = 0 THEN
      IF ⋆(n)
      THEN RETURN(True);
      ELSE RETURN(False);
2.  FOREACH n' IN successors(n) DO
      IF solved-labeling(G, n', successors, ⋆)
      THEN
        IF OR_node(n) THEN RETURN(True); // One success is enough.
      ELSE
        IF AND_node(n) THEN RETURN(False); // One fail is one too much.
    ENDDO
3.  IF OR_node(n)
    THEN RETURN(False); // OR_node n has no solvable successor.
    ELSE RETURN(True); // All successors of AND_node n are solvable.
```

Problem-Reduction Representation

Algorithm: solved-labeling (with labeling)

Input: G. A finite, acyclic AND-OR graph.
n. A node in G.
successors(n). Returns the successors of node n.
$\star(n)$. Predicate that is *True* if n is a goal node.

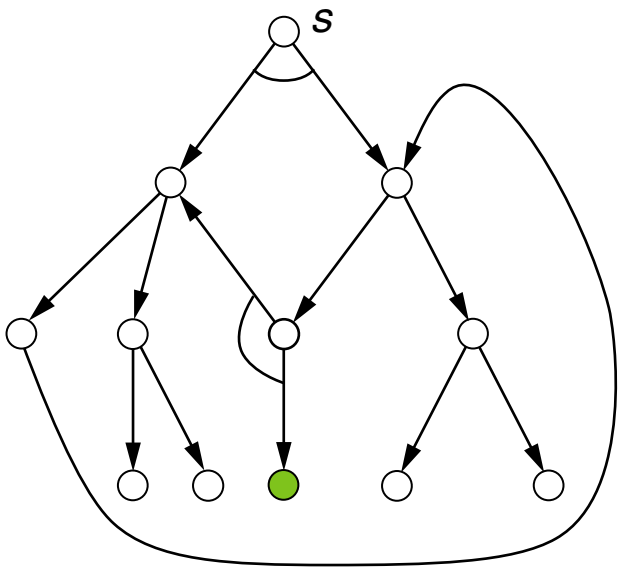
Output: The symbol *True* if a solution exists, *False* otherwise.

```
solved-labeling(G, n, successors, ⋆)

1.  IF solved(n) THEN RETURN(True);

2.  IF |successors(n)| = 0 THEN
      IF ⋆(n)
      THEN solved(n) = True, RETURN(True);
      ELSE RETURN(False);

3.  FOREACH n' IN successors(n) DO
      IF solved-labeling(G, n', successors, ⋆)
      THEN
        IF OR_node(n) THEN solved(n) = True, RETURN(True);
      ELSE
        IF AND_node(n) THEN RETURN(False);
    ENDDO

4.  IF OR_node(n)
    THEN RETURN(False);
    ELSE solved(n) = True, RETURN(True);
```

Problem-Reduction Representation

Solution Graphs with Cycles (1)

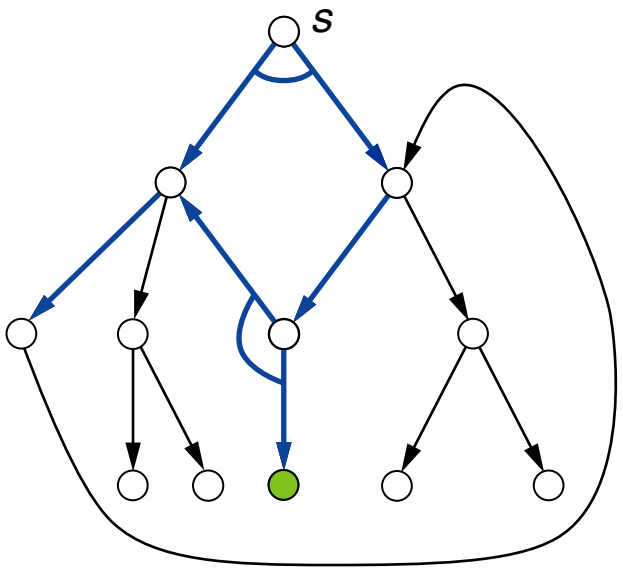
Cyclic problem-reduction graphs can entail infinite solution graphs, caused by “unfolding”:



Problem-Reduction Representation

Solution Graphs with Cycles (1)

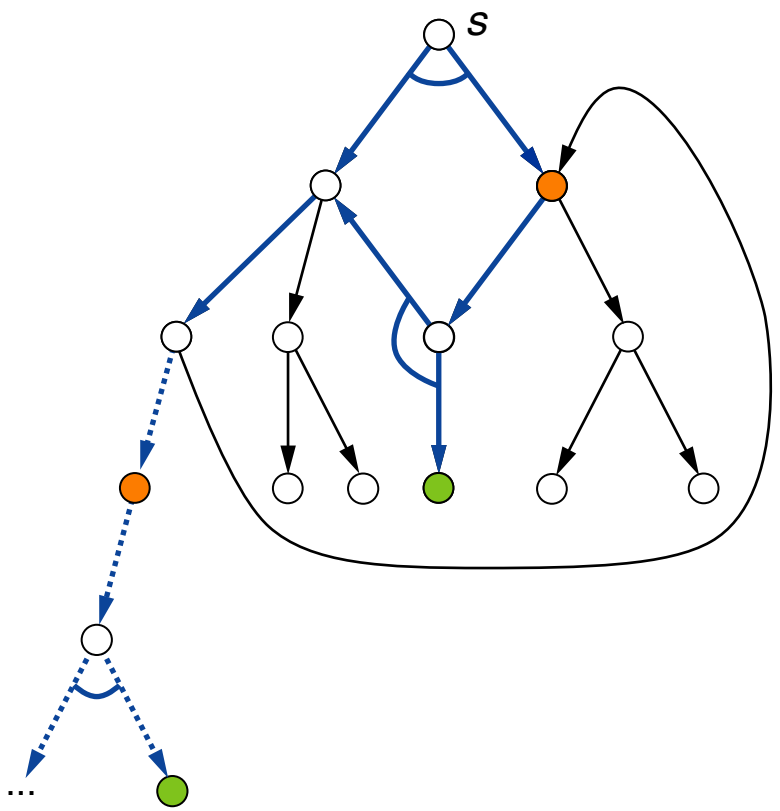
Cyclic problem-reduction graphs can entail infinite solution graphs, caused by “unfolding”:



Problem-Reduction Representation

Solution Graphs with Cycles (1)

Cyclic problem-reduction graphs can entail infinite solution graphs, caused by “unfolding”:



Remarks:

- "Unfolding" means that when a node is reached a second time, the state is not reused but a copy is created instead. I.e., we are completely omitting an occur check.

- Note that the solution graph definition restricts to acyclic AND-OR graphs.
  Q. How could the solved-labeling procedure be adapted to deal with cyclic AND-OR graphs?

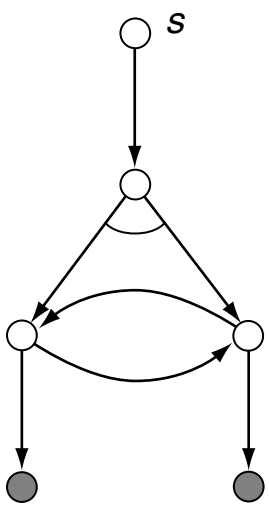
- Usually the search space underlying a problem is only partially explored, and hence the AND-OR graph H does not show the complete picture.
  Q. How could the solved-labeling procedure be adapted to deal with incomplete AND-OR graphs?

- Q. How could an "unsolvable-labeling procedure" be defined, which proves whether a problem is unsolvable?

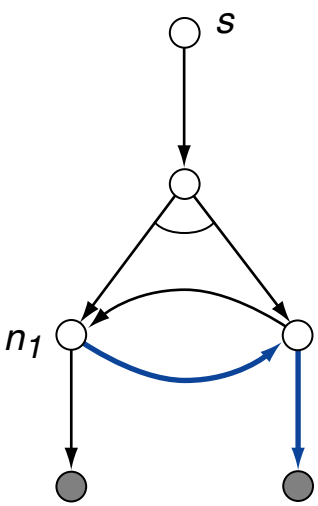
Problem-Reduction Representation

Solution Graphs with Cycles (2)

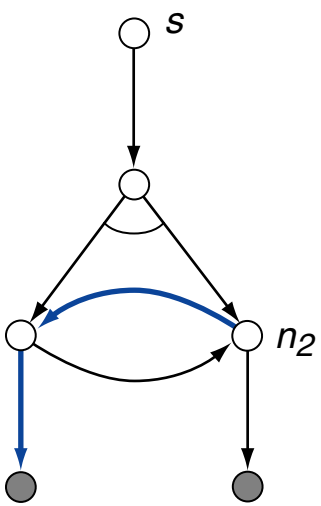
Cyclic problem-reduction graphs can entail cyclic synthesized solution graphs even if an acyclic solution graph exists:



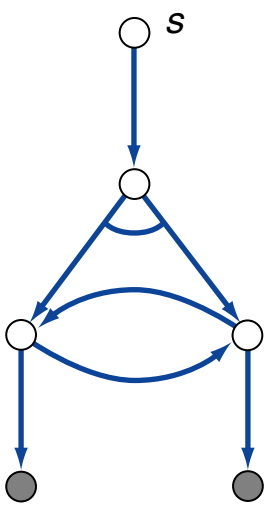
AND-OR graph



Solution graph for n_1



Solution graph for n_2

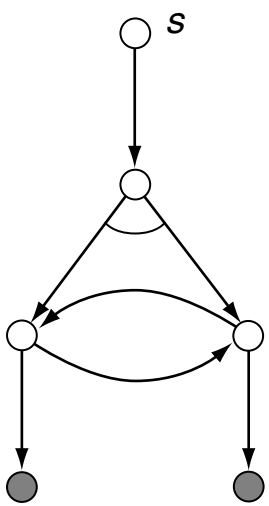


Solution graph for s

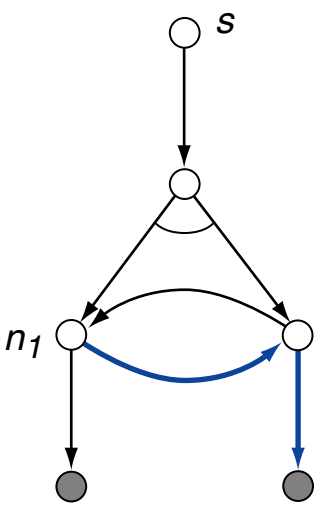
Problem-Reduction Representation

Solution Graphs with Cycles (2)

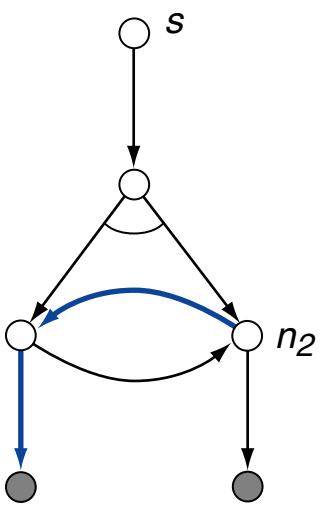
Cyclic problem-reduction graphs can entail cyclic synthesized solution graphs even if an acyclic solution graph exists:



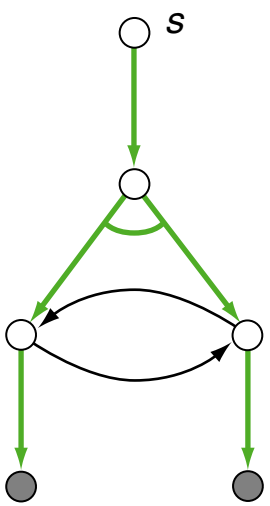
AND-OR graph



Solution graph for n_1



Solution graph for n_2



Acyclic solution graph for s

Problem-Reduction Representation

Hypergraphs

AND-OR graphs can be considered as a generalization of ordinary graphs called *hypergraphs*:

- Ordinary graph: Directed links (edges, arcs) connect two nodes.
- Hypergraph: Directed hyperedges, also called “connectors”, connect two *sets* of nodes.
- AND-OR graph: The links of an AND node form a single hyperedge.

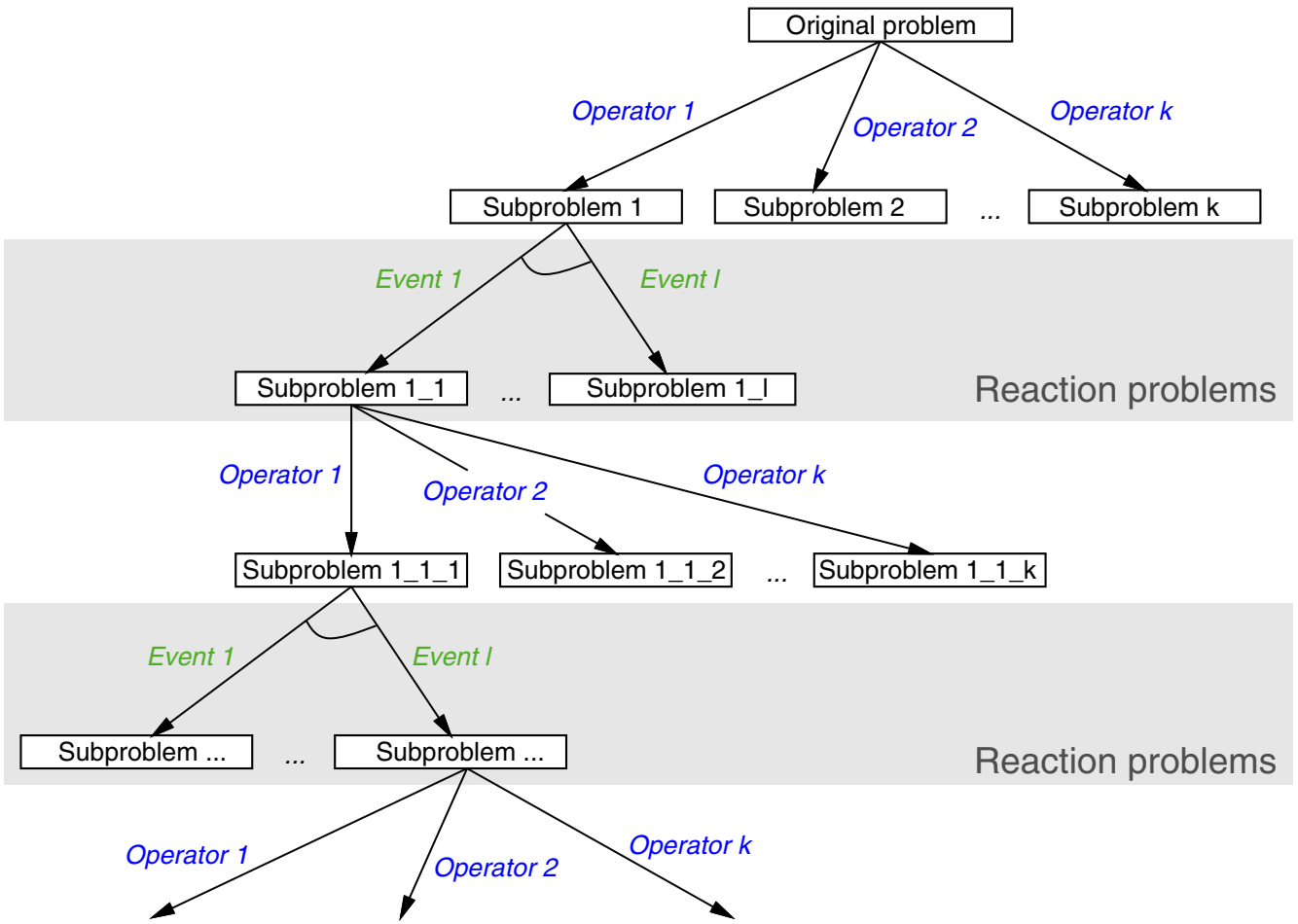
➜ The determination of a solution graph corresponds to the determination of a hyperpath between the root node s and a set of goal nodes that represent solved rest problems.

Remarks:

- The hyperedges in an AND-OR graph are also called F-edges. [Gallo 1992]
- The definition of hyperpaths includes the property of being minimal.
  [Nielsen/Andersen/Pretolani 2005]

Choosing a Representation

Problem-Reduction Graphs with Alternating Node Types



Choosing a Representation

Problem-Reduction Graphs with Alternating Node Types (continued)

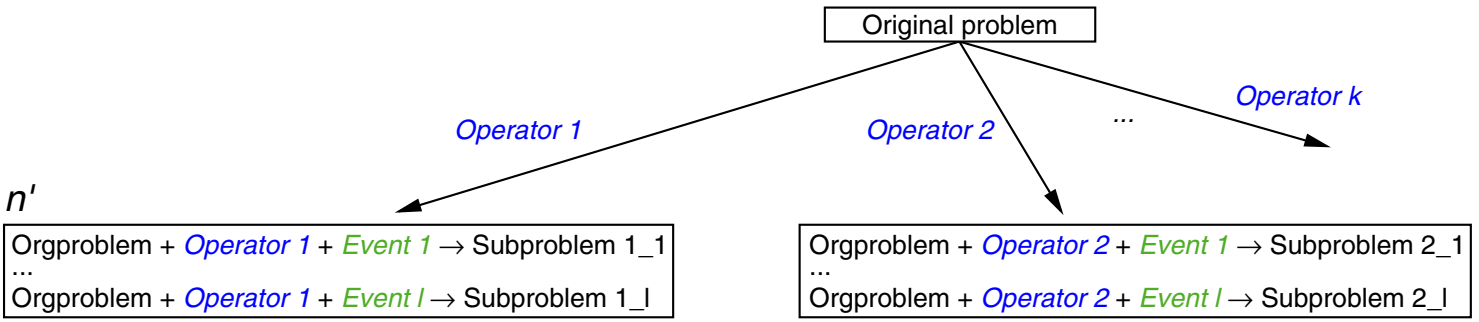
Interpretation:

- k operators $\sim$ own actions, decisions, strategy
- l events $\sim$ reactions

- Action problems.
  Consequences that result from a subproblem under the impact of applicable operators.
- Reaction problems.
  Consequences that result from a subproblem under the impact of possible events.

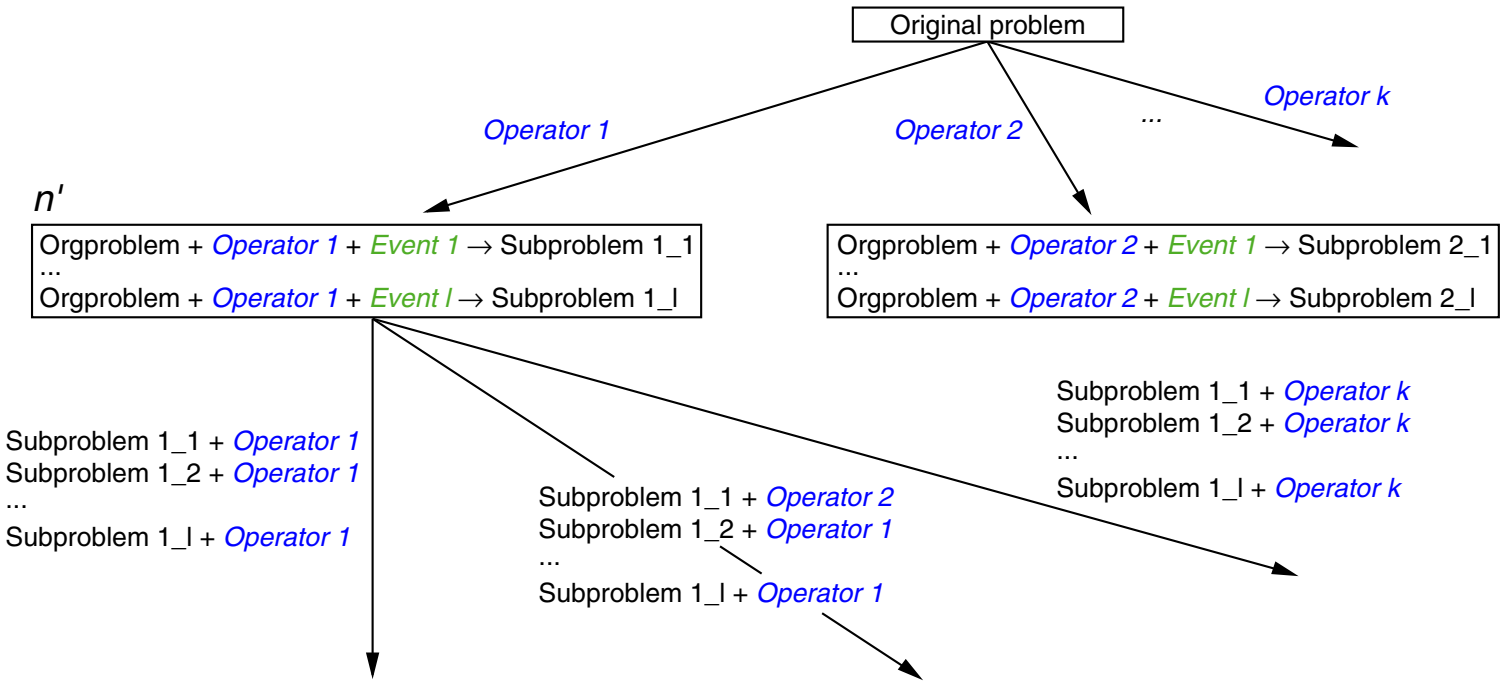
Choosing a Representation

Transforming Problem-Reduction into State-Space Graphs



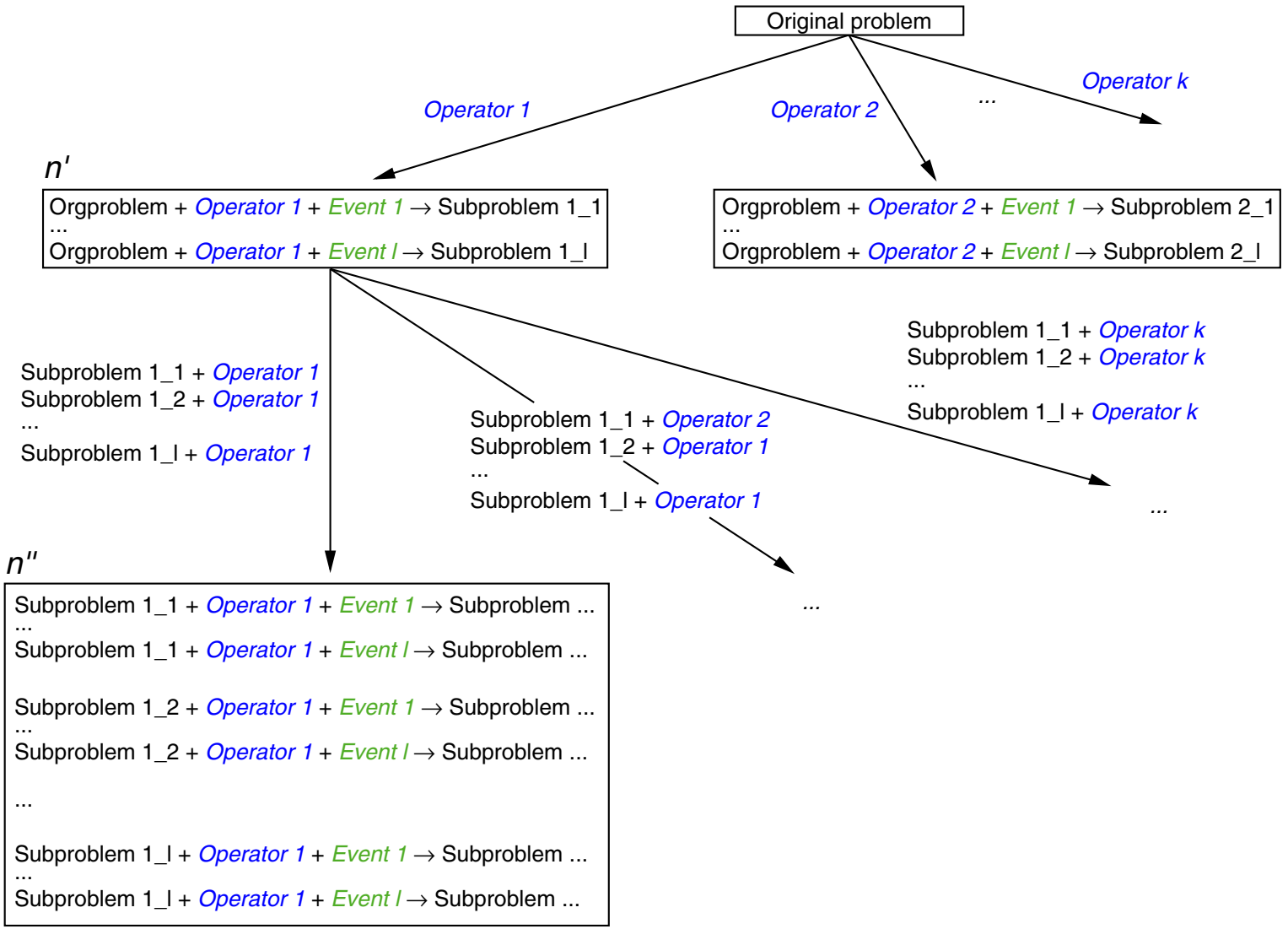
Choosing a Representation

Transforming Problem-Reduction into State-Space Graphs



Choosing a Representation

Transforming Problem-Reduction into State-Space Graphs



Choosing a Representation

Transforming Problem-Reduction into State-Space Graphs (continued)

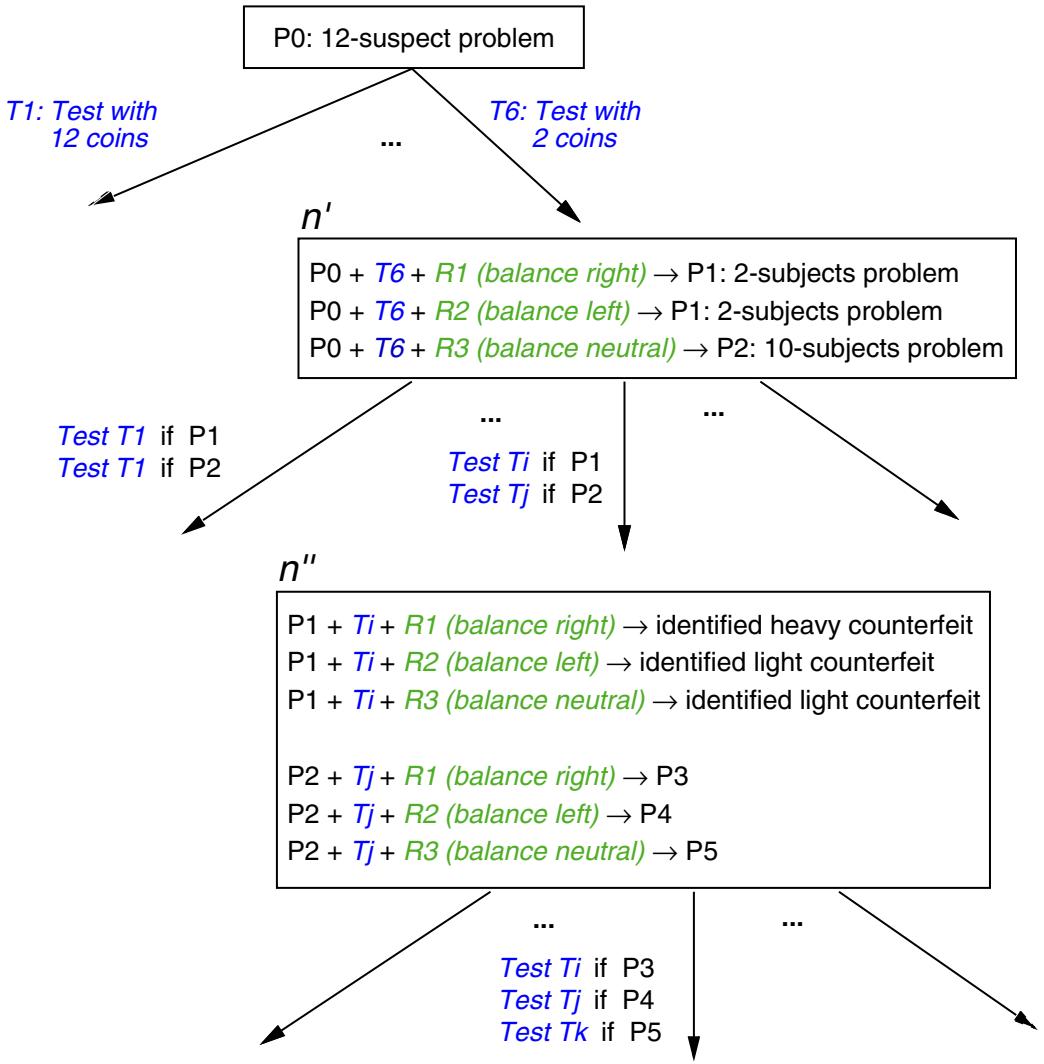
Transformation rules:

- To the node s the applicable operators are applied.
- The successor node n' of s is comprised of all reaction problems that may result from applying the chosen operator.
- To the node n' all applicable operator combinations are applied, whereas each subproblem $p \in n'$ gets its own operator, i.e., a set of $|n'|$ operators is associated with the link (n', n'').
- The successor node n'' of n' is comprised of all reaction problems that may result from applying the chosen operator set.

  Observe that the node n' has as many successors as operator combinations are possible, given the subproblems $p \in n'$. This number is in $O(k^{|n'|})$

Choosing a Representation

Transformed Counterfeit Problem



Choosing a Representation

State-Space versus Problem-Reduction Representation

A state-space representation is advisable if solutions are paths (or nodes).

- ➜ The solution graph is a path in the state-space graph.

Examples:

- constraint satisfaction problems
- sequence seeking problems

Choosing a Representation

State-Space versus Problem-Reduction Representation (continued)

A problem-reduction representation is advisable if solution are trees, graphs, or partially ordered node sets.

➜ The solution graph is a tree-like subgraph of the problem-reduction graph.

Examples:

- strategy seeking problems
- symbolic integration
- theorem proving

Characteristics:

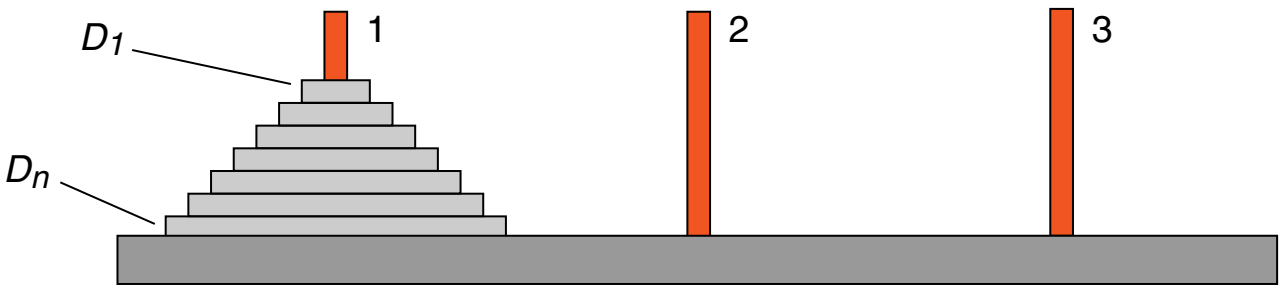
- Problems may be decomposed into subproblems that can be solved independently of each other (= in parallel).
- The divide-and-conquer paradigm can be applied to find a global optimum.

Remarks:

- Choosing a search space representation requires the analysis whether and how the solutions of subproblem interact with each other when being composed.
- Q. Is a problem-reduction representation possible for the 8-puzzle problem?
- Q. Is a problem-reduction representation an advisable choice for the 8-puzzle problem?
- There are sequence seeking problems where the solutions of subproblems interact with each other (e.g. the 8-puzzle problem), but where a problem-reduction representation is superior to a state-space representation. Example: The tower of Hanoi problem.

Choosing a Representation

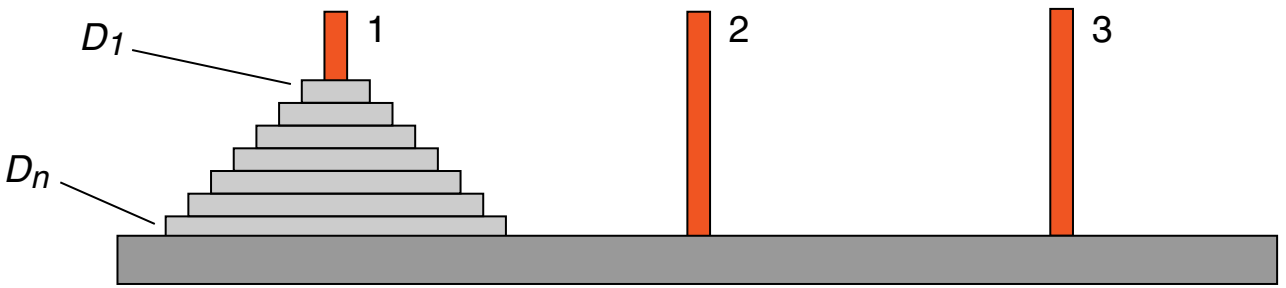
Tower of Hanoi Problem [Wikipedia]



A stack of n disks $D_1, \ldots, D_n$ (different sizes, sorted) is to be moved from peg 1 to peg 3. Only one disk can be moved at a time, no disk may he placed on top of a smaller one.

Choosing a Representation

Tower of Hanoi Problem [Wikipedia]



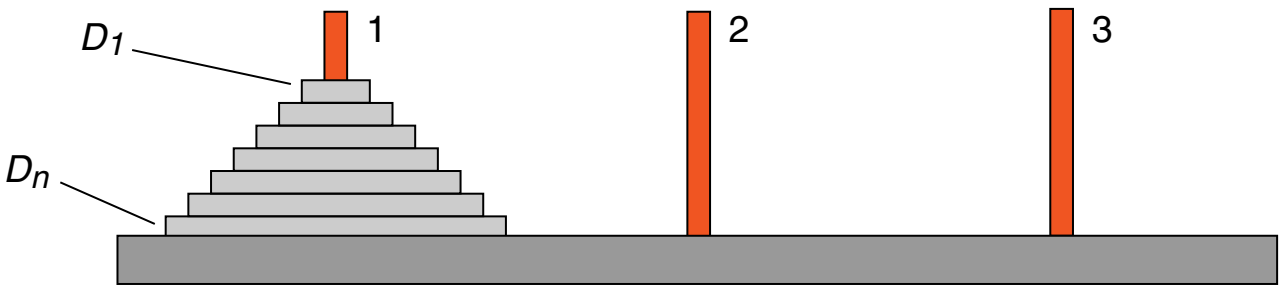
A stack of n disks $D_1, \ldots, D_n$ (different sizes, sorted) is to be moved from peg 1 to peg 3. Only one disk can be moved at a time, no disk may he placed on top of a smaller one.

Observations:

- The n subproblems (move $D_i, i = 1, \ldots, n$ on peg 3) are not independent. A solution based on a state-space representation may be advisable.
- There is a linear connection between the subproblems: as soon as $D_{k+1}, \ldots, D_n$ are moved, $D_1, \ldots, D_k$ can be moved as well without moving $D_{k+1}, \ldots, D_n$.

Choosing a Representation

Tower of Hanoi Problem [Wikipedia]



A stack of n disks $D_1, \ldots, D_n$ (different sizes, sorted) is to be moved from peg 1 to peg 3. Only one disk can be moved at a time, no disk may he placed on top of a smaller one.

Observations:

- The n subproblems (move $D_i, i = 1, \ldots, n$ on peg 3) are not independent. A solution based on a state-space representation may be advisable.
- There is a linear connection between the subproblems: as soon as $D_{k+1}, \ldots, D_n$ are moved, $D_1, \ldots, D_k$ can be moved as well without moving $D_{k+1}, \ldots, D_n$.

➜ Start with the most difficult problem: D_n from peg 1 to peg 3.

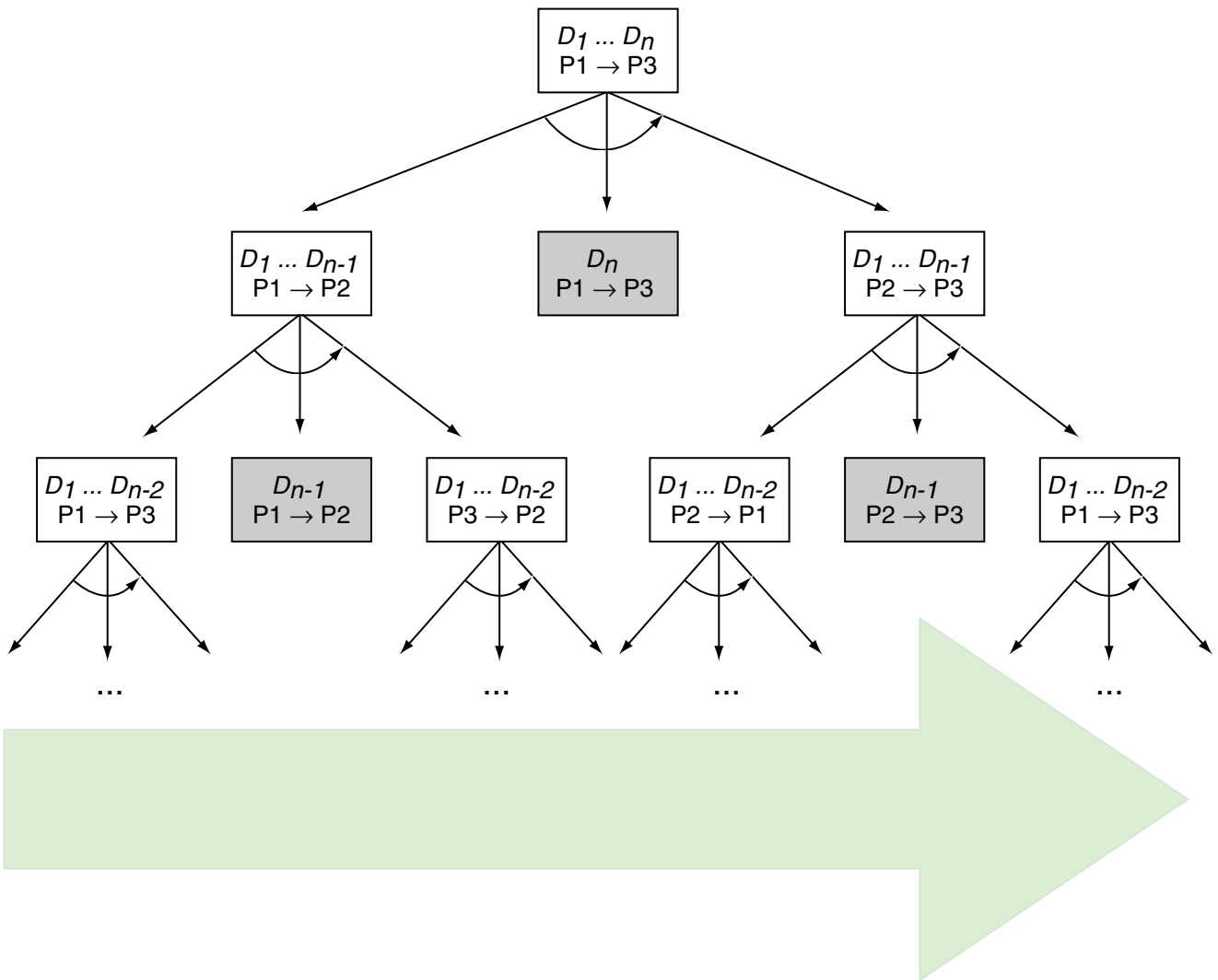
➜ D_n must be clear and peg 3 must be empty.

➜ Introduce new subproblems: $D_1, \ldots, D_{n-1}$ from peg 1 to peg 2.

➜ Solve the remaining problems.

Choosing a Representation

Tower of Hanoi Problem (continued)



Choosing a Representation

Means-End Analysis

Characteristic for the tower of Hanoi problem is that the set of subproblems $p_1, \ldots, p_n$ is serializable. Formally, this means that there is a sequence $i_1, \ldots, i_n$ such that for problems $p_{i_1}, \ldots, p_{i_n}$ hold:

1. $p_{i_1}, \ldots, p_{i_k}$ can be solved after $p_{i_{k+1}}, \ldots, p_{i_n}$ have been solved.
2. Solving $p_{i_1}, \ldots, p_{i_k}$ will not compromise the solutions of $p_{i_{k+1}}, \ldots, p_{i_n}$.

Choosing a Representation

Means-End Analysis

Characteristic for the tower of Hanoi problem is that the set of subproblems $p_1, \ldots, p_n$ is serializable. Formally, this means that there is a sequence $i_1, \ldots, i_n$ such that for problems $p_{i_1}, \ldots, p_{i_n}$ hold:

1. $p_{i_1}, \ldots, p_{i_k}$ can be solved after $p_{i_{k+1}}, \ldots, p_{i_n}$ have been solved.
2. Solving $p_{i_1}, \ldots, p_{i_k}$ will not compromise the solutions of $p_{i_{k+1}}, \ldots, p_{i_n}$.

If this underlying problem structure is identified, the powerful operator selection strategy "means end analysis" can be applied:

> *"The basic difference between this method* [means end analysis] *and the state-space approach is its purposeful behavior: operators are invoked by virtue of their potential in fulfilling a desirable subgoal, and new subgoals are created in order to enable the activation of a desirable operator."*
>
> [Pearl 1984, p.29]

Remarks:

- Planning a trip (e.g. to New York) is another example for a problem that can be tackled by a means end analysis:
  1. New York $\rightarrow$ plane from Frankfurt
  2. plane from Frankfurt $\rightarrow$ train to Frankfurt
  3. train to Frankfurt $\rightarrow$ bus to train station
  4. ...
- The power of problem reduction representation under a regime of a means end analysis: AND-linked subproblems can be solved independently, though for processing the global plan a strict order (from left to right) must be obeyed.
- Q. What is the complexity class of the tower of Hanoi problem?