Artificial Intelligence: And-Tree-based Search

CPSC 433: Artificial Intelligence
Fall 2022

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Thursday, September 29, 2022

UNIVERSITY OF CALGARY

And-tree-based Search

Basic Idea:

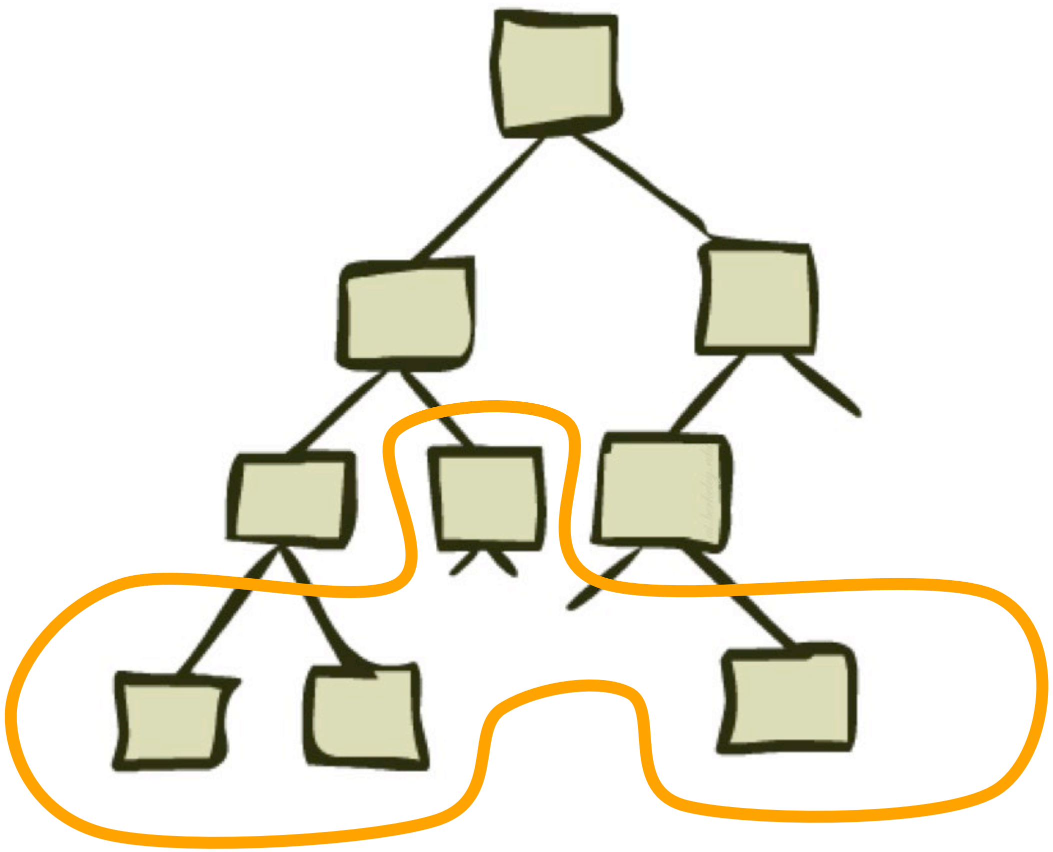
1. Divide a problem into subproblems, whose solutions can be put together into a solution for the initial problem.

Examples of subproblem division:

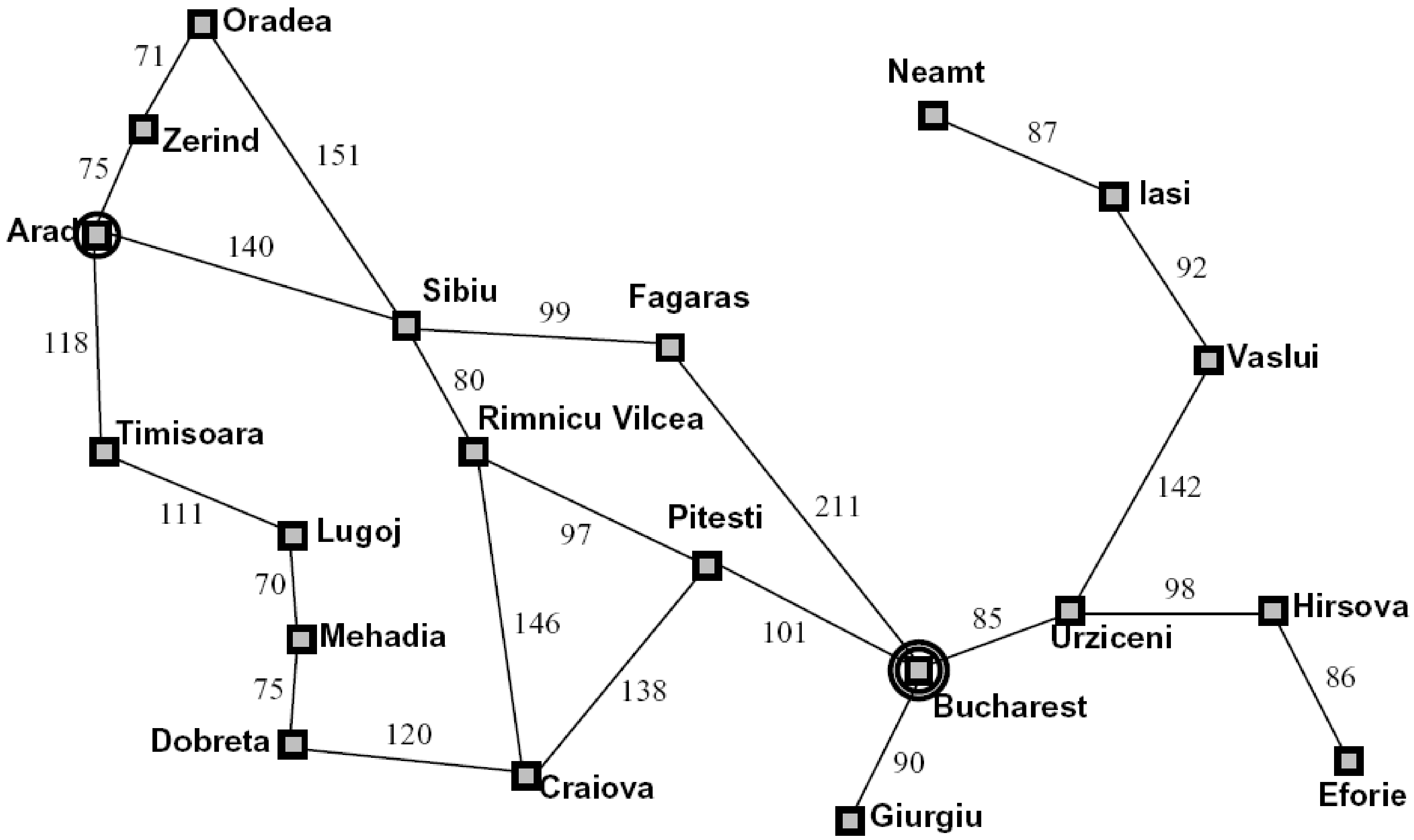
- Construction of something: different parts of it
- Optimization problems: different instantiations of free variables; putting solution together by comparing all possibilities

Tree Search

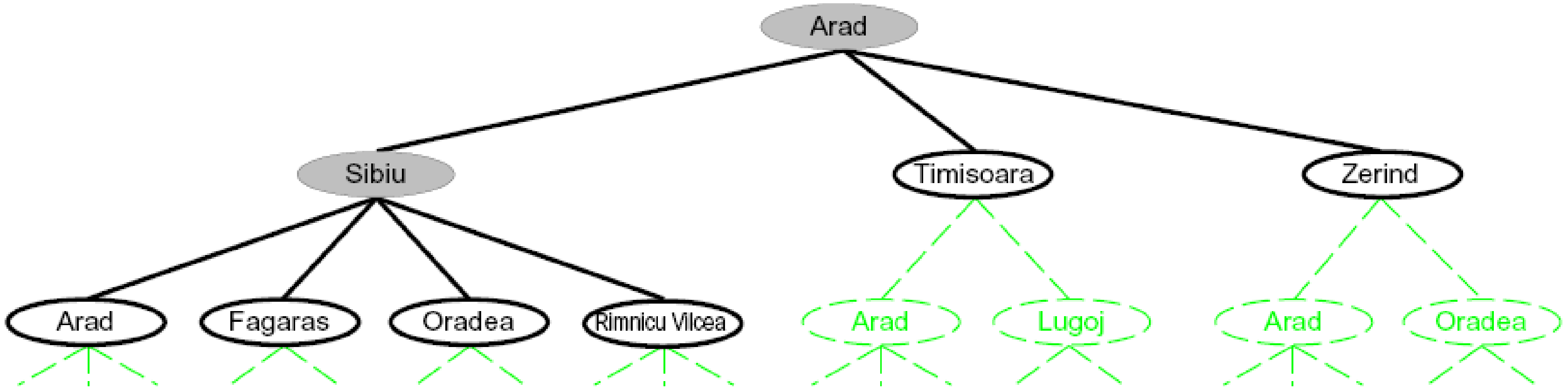
Tree Search



Search Example: Romania



Searching with a Search Tree



- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

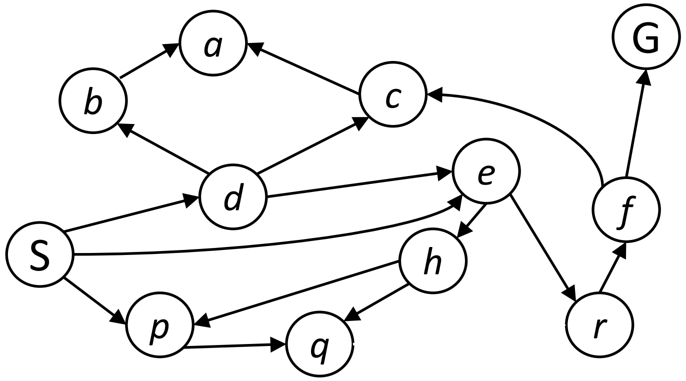
General Tree Search

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

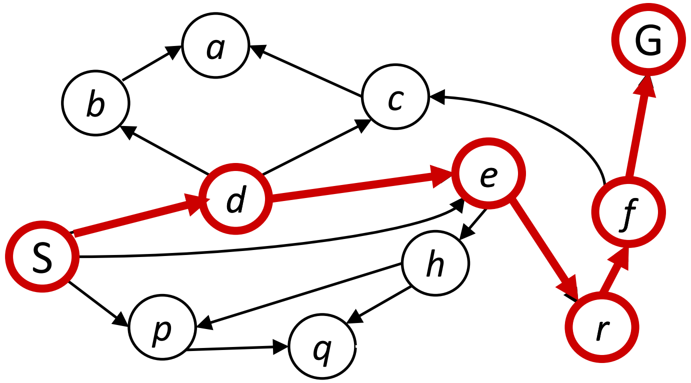
- Important ideas:
  - Fringe
  - Expansion
  - Exploration strategy

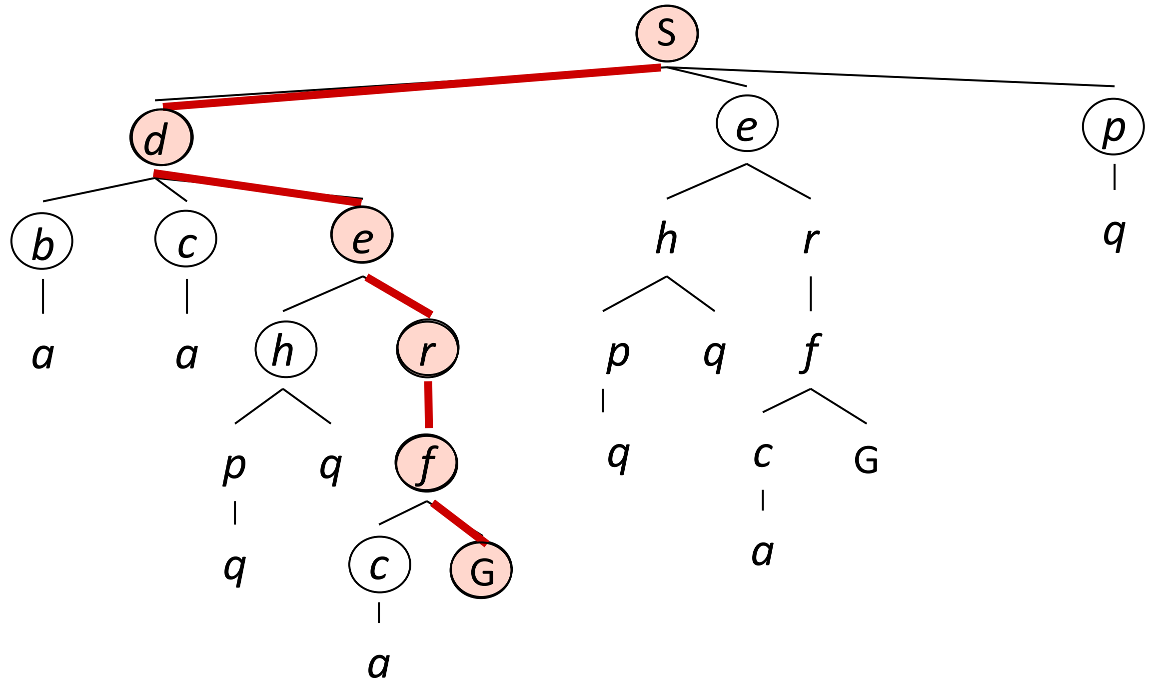
- Main question: which fringe nodes to explore?

Example: Tree Search



Example: Tree Search





~~s~~

~~s → d~~

s → e

s → p

s → d → b

s → d → c

~~s → d → e~~

s → d → e → h

~~s → d → e → r~~

~~s → d → e → r → f~~

s → d → e → r → f → c

~~s → d → e → r → f → G~~

Definitions

And-tree (one type of tree search)

Formal Definitions: Model

And-tree-based Search Model $A_\wedge = (S_\wedge, T_\wedge)$

$Prob$ set of problem descriptions

$Div \subseteq Prob^+$ division relation ($Prob^+ \rightarrow$ things that can be generated by dividing problems in $Prob$)

$S_\wedge \subseteq Atree$ set of possible states, is subset tree structures

where $Atree$ is recursively defined by

$(pr, sol) \in Atree$ for $pr \in Prob, sol \in \{yes, ?\}$

$(pr, sol, b_1, \ldots, bn) \in Atree$ for $pr \in Prob, sol \in \{yes, ?\}$, $b_i \in Atree$

$T_\wedge \subseteq S_\wedge \times S_\wedge$ transitions between states, but more specifically

$T_\wedge = \{(s_1, s_2) \mid s_1, s_2 \in S_\wedge$ and $Erw_\wedge(s_1, s_2)$ or $Erw_\wedge^*(s_1, s_2)\}$

Less formally: Model

- $Prob$ usually is described using an additional data structure: a set of formulas describing the world, a matrix describing distances to remaining cities, and so on.
- $Prob$ can also just remember all decisions made so far
- Obviously, different problems produce different sets $Prob$
- Div formally describes what divisions of problems into subproblems are possible; also absolutely dependent on the problem we want to solve.

Less formally: Model (II)

- A node containing a problem and a sol-entry is an **and-tree** (*Atree*).
- If we have several (i.e. n) **and-trees**, then putting them as successors to a node representing a problem and a sol-entry also produces an **and-tree**.

Note: this does not say anything about the connection between the problems in such a tree; in fact, most elements of $Atree$ will never be used as search states, because they do not make sense for the application.

Formal Definitions: Erw (Extension function)

$Erw_\wedge$ and $Erw_\wedge^*$ are relations on $Atree$ defined by

- $Erw_\wedge\big((pr,?),(pr,yes)\big)$ if pr is solved
- $Erw_\wedge\Big((pr,?),\big(pr,?,(pr_1,?),\ldots,(pr_n,?)\big)\Big)$ if $\mathrm{D}iv(pr,pr_1,\ldots,pr_n)$ holds
- $Erw_\wedge\big((pr,?,b_1,\ldots,b_n),(pr,?,b_1',\ldots,b_n')\big)$

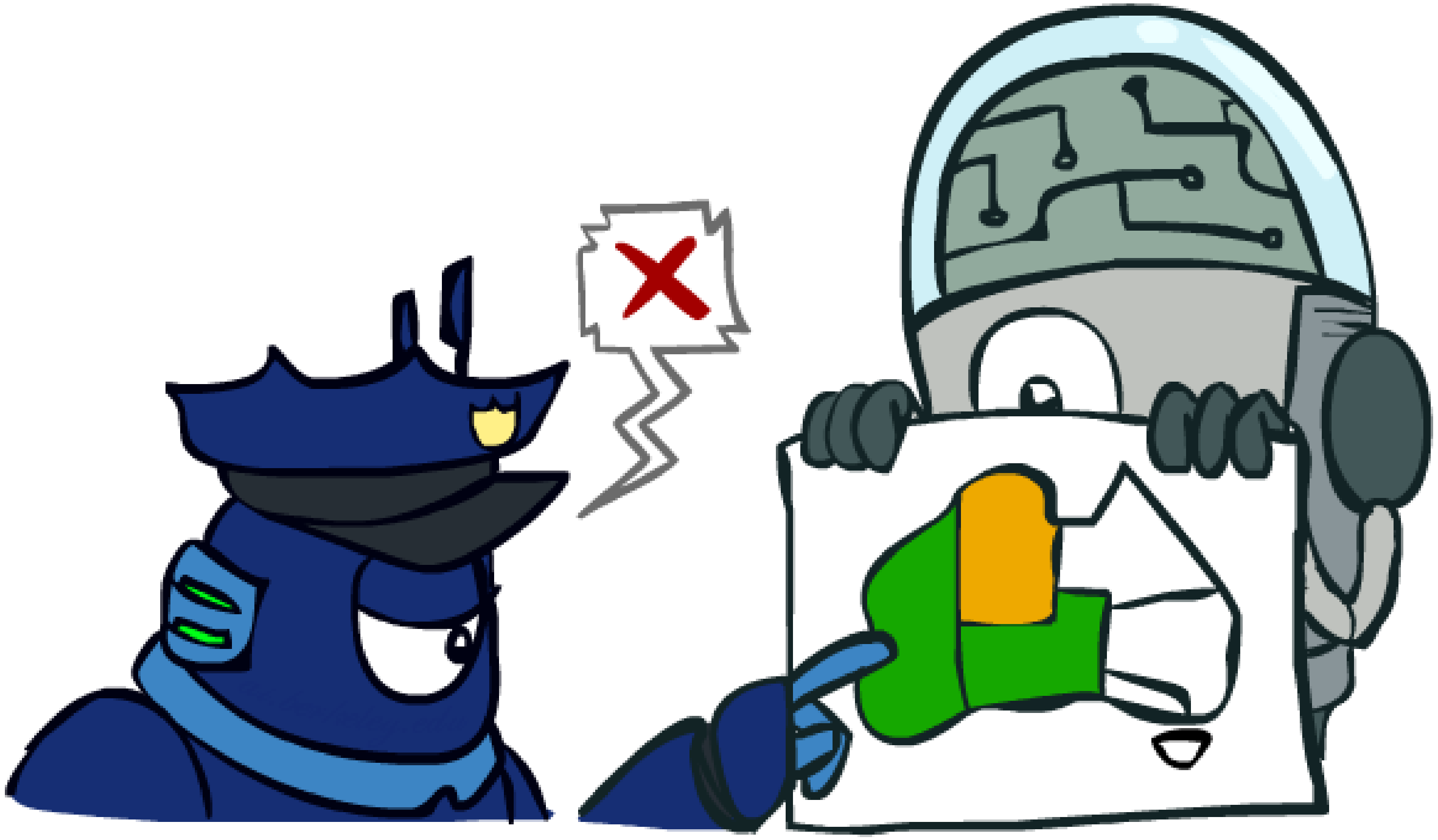
  if for an i: $Erw_\wedge(b_i,bi')$ and $\mathrm{b}_j = b_j{}'$ for $i \neq j$
- $Erw_\wedge \subseteq Erw_\wedge^*$
- $Erw_\wedge^*\big((pr,?,b_1,\ldots,b_n),(pr,?,b_1{}',\ldots,b_n{}')\big)$

  if for all i either $Erw_\wedge^*(b_i,bi')$ or $\mathrm{b}_i = b_i{}'$ holds

Less formally: Erw (Extension function)

- $Erw_\wedge$ connects and-trees that reflect the idea of dividing problems into subproblems
  - if we know the solution to a problem in a node (i.e. it is solved for us), we mark it (sol-entry yes)
  - else, if we know the division of a problem in a (leaf) node into subproblems, then we generate successors to this node for each subproblem
  - else, see remarks about $Erw_\wedge^*$
- The 3rd definition for $Erw_\wedge$ allows us to apply the construction of above not only to a root node, but to leaf nodes of a tree.

Backtracking Search



Less formally: Erw* (Extension function)

- $Erw_{\wedge}^{*}$ is for intelligent backtracking (note the sequence of arguments in the definition of $T_{\wedge}$). It allows us to take away the results of several applications of $Erw_{\wedge}$ as one transition (therefore "intelligent").
- Backtracking is necessary, if you reach a tree with a leaf that neither represents a solved problem nor has a problem that can be divided into subproblems (or we already have unsuccessfully tried out all of its divisions defined by Div).
- Controls usually employ backtracking only in very clearly defined (special) cases.

Formal Definitions: Search Process

And-tree-based Search Process $P_\wedge = (A_\wedge, Env, K_\wedge)$

Not more specific than general definition given previously

But: often control uses two functions

- one function f_{leaf} that compares all leaves of the tree representing the state and selecting one
- one function f_{trans} that selects one of the transitions that deal with the selected leaf

UNIVERSITY OF CALGARY

Less formally: Search Process

- Due to the possibility of having several divisions of the same problem in Div, first determining a leaf to "expand" and then selecting the division is often sensible.
- But sometimes the availability of certain divisions determines what leaf to select next, so that f_{leaf} and f_{trans} are not always used.
- An and-tree-based search starts with putting the problem instance to solve into the root of an and-tree.
- If we have found a solution to every subproblem represented by a leaf, then it is still possible that the solutions are not compatible. Then other solutions have to be found (☞ backtracking).

Formal Definitions: Search Instance (IV)

And-tree-based Search Instance $Ins_\wedge = (s_0, G_\wedge)$

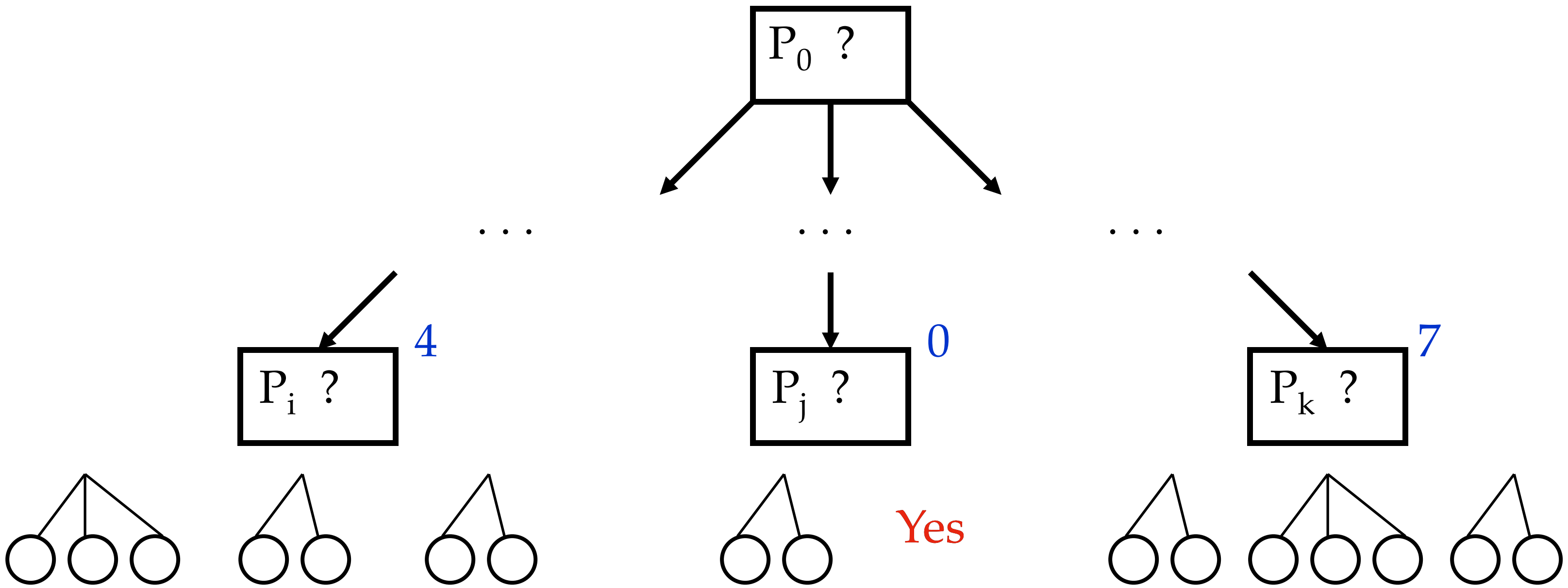
If the given problem to solve is pr, then we have

- $s_0 = (pr, ?)$
- $G_\wedge(s) = yes$, if and only if
  1. $s = (pr', yes)$ or
  2. $s = (pr', ?, b_1, \ldots, bn), G_\wedge(b_1) = \cdots = G_\wedge(bn) = yes$ and the solutions to $b_1, \ldots, bn$ are compatible with each other or
  3. there is no transition that has not been tried out already

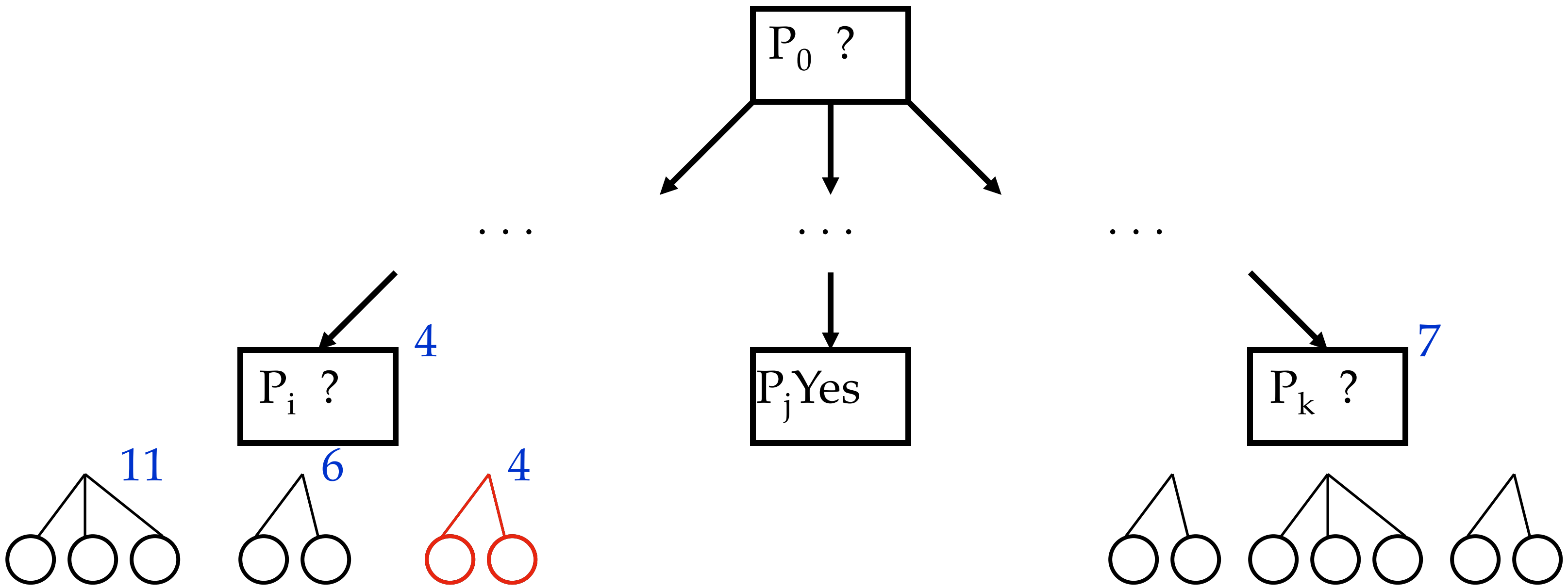
UNIVERSITY OF CALGARY

Visualize

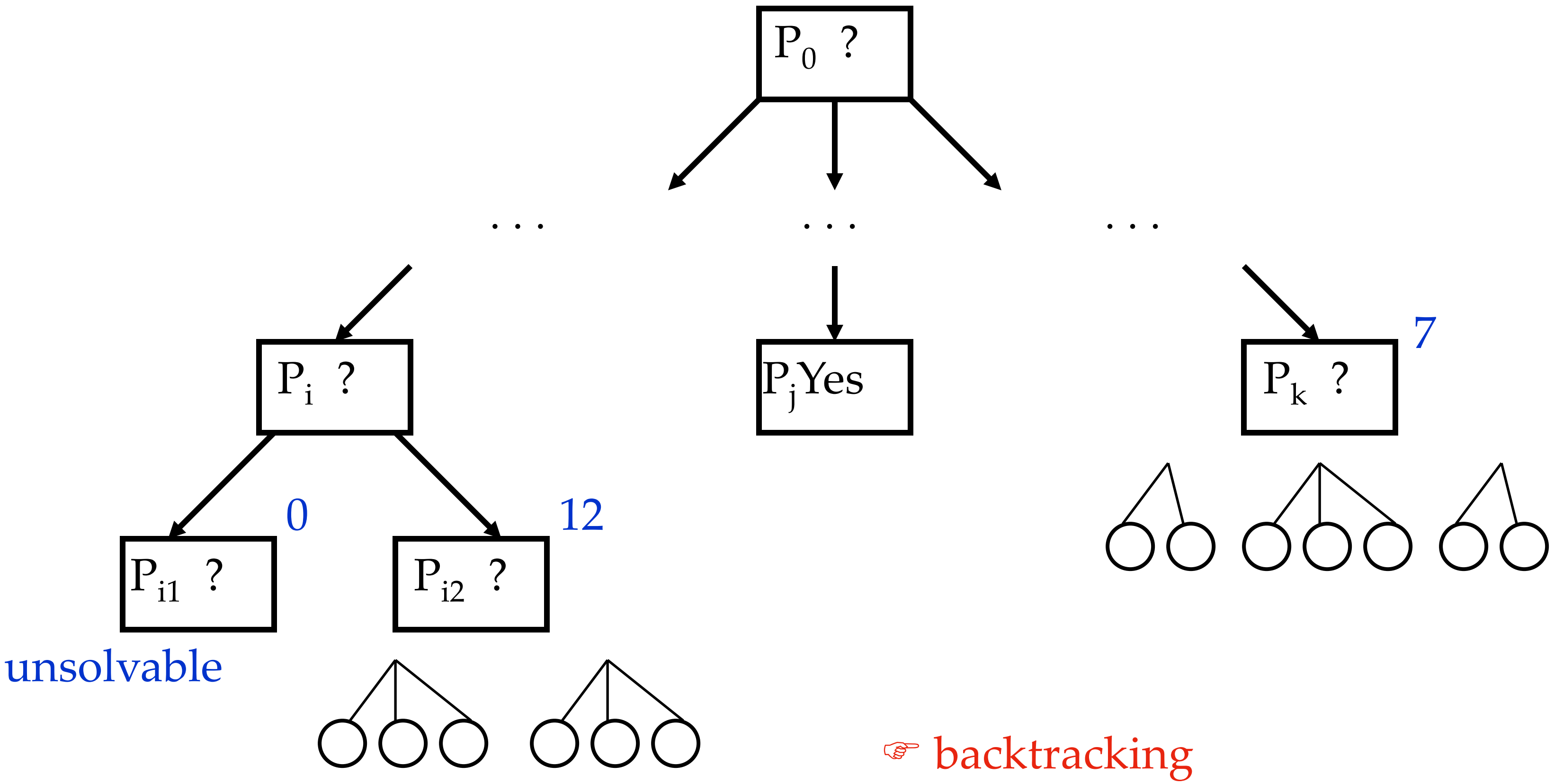
Conceptual Example (II): And-tree-based Search



Conceptual Example (II): And-tree-based Search

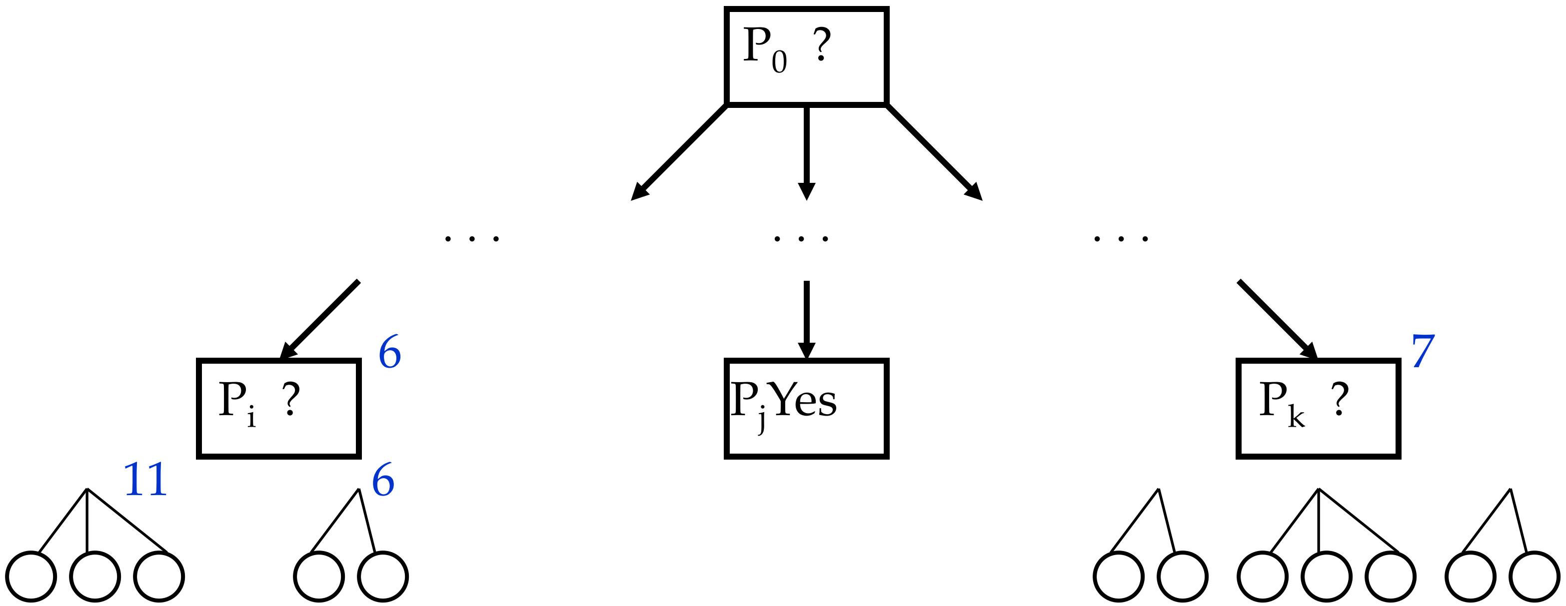


Conceptual Example (II): And-tree-based Search



☞ backtracking

Conceptual Example (II): And-tree-based Search



Design

Designing and-tree-based search models

1. Identify how you can describe a problem (resp. what is needed to describe sub-problems) ☞ $Prob$
2. Define how to identify if a problem is solved
3. Identify the basic ideas how to divide a problem into subproblems ☞ Div
4. Determine if it is possible that you run into deadends (i.e. can there be leafs that neither are solved nor appear in Div as first argument). If yes, we need backtracking, if no, we do not need backtracking.

Designing and-tree-based search processes

1. Identify how you can measure a problem in a leaf
   1. Priority to problems that are solved
   2. See other slides for criteria
2. Use 1. to come up with a f_{leaf}-function comparing the leaves in an and-tree.
3. For the f_{trans}-function that determines the transition you are doing:
   1. If there is an unsolvable problem in a leaf then backtrack
   2. If the selected leaf can be solved, do it
   3. Determine the different divisions of the leaf problem and measure them

Applied to Model-elimination

Concrete Example: Model-elimination

- Another, now analytical, way to solve the problem of determining if a formula is a consequence of a set of formulas
- Again works with sets of clauses
- A problem is divided into subproblems by employing a clause $L_1 \vee \ldots \vee L_n$: n subproblems are generated, each of which assumes that additionally a certain instance σ of L_i is true (each subproblem uses a different L_i but the same σ)

Modelelimination (II)

- We start with a "world" containing no predicate or its negation (i.e. everything is possible)
- Then we select a leaf in our tree and a clause
  $L_1 \vee \ldots \vee L_n$ and generate the successor nodes as described above.
  One additional condition is that at least one of the resulting subproblems is solved (except for a transition out of the "empty" world).
- A subproblem is solved, if it contains P and $\neg$P' such that there is a σ with $\sigma(P) \equiv \sigma(P')$ (usually we use $\sigma = mgu(P,P')$)

Modelelimination (III)

- By using the mgu, each time we do this, we have to apply it to all subproblems we have generated so far (in order to guarantee that solutions to subproblems are compatible).
- Our problem is solved (positively), if all subproblems are solved.

Model-elimination: Examples

Modelelimination (IV)

- Solve the following problem instances:

1) $p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$
2) $p, q, \neg q$
3) $P(x) \lor R(x), \neg R\big(f(a,b)\big), \neg P(g(a,b))$

Modelelimination (IV)

- Solve the following problem instances:

$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

Modelelimination (IV)

- Solve the following problem instances:

$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

$(\{\}, ?)$

UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

$\boldsymbol{p} \vee \boldsymbol{q}, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q$

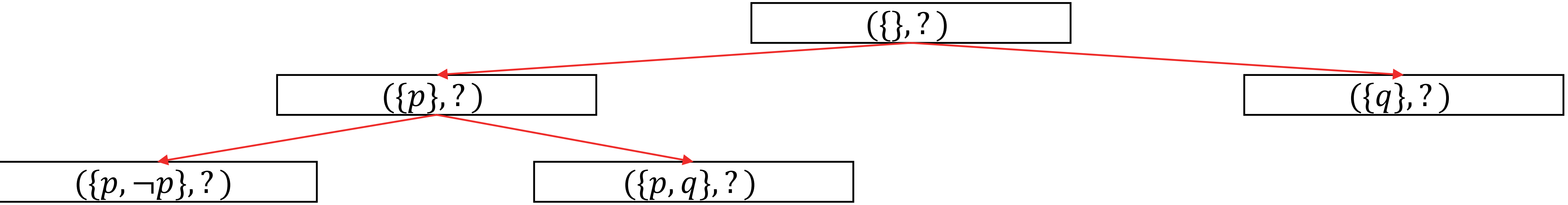
$(\{\}, ?)$

$(\{p\}, ?)$ $(\{q\}, ?)$

Modelelimination (IV)

- Solve the following problem instances:

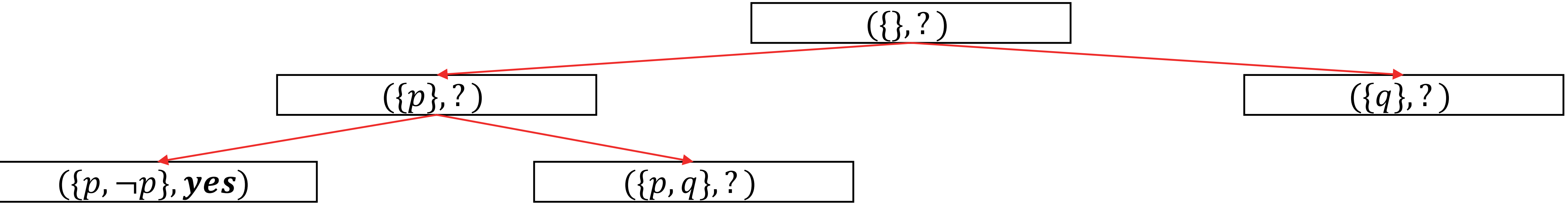
$p \lor q, p \lor \neg q, \neg \boldsymbol{p} \lor \boldsymbol{q}, \neg p \lor \neg q$



Modelelimination (IV)

- Solve the following problem instances:

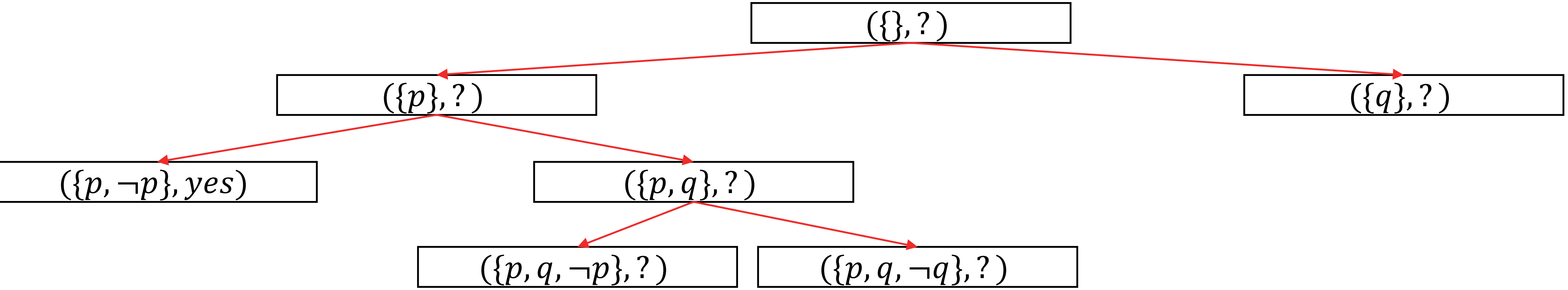
$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$



Modelelimination (IV)

- Solve the following problem instances:

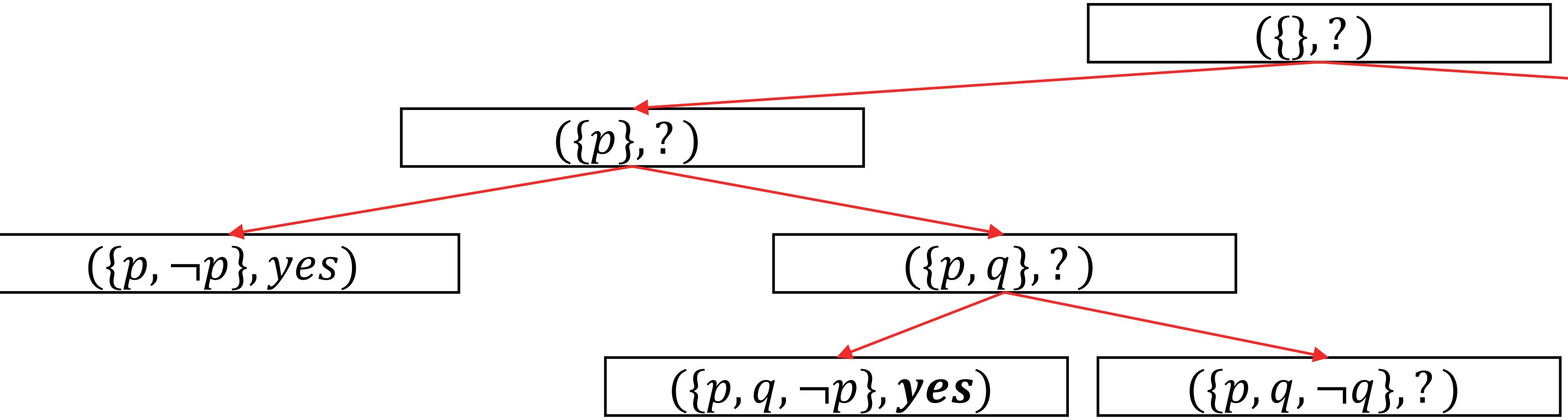
$p \vee q, p \vee \neg q, \neg p \vee q, \neg \boldsymbol{p} \vee \neg \boldsymbol{q}$



Modelelimination (IV)

- Solve the following problem instances:

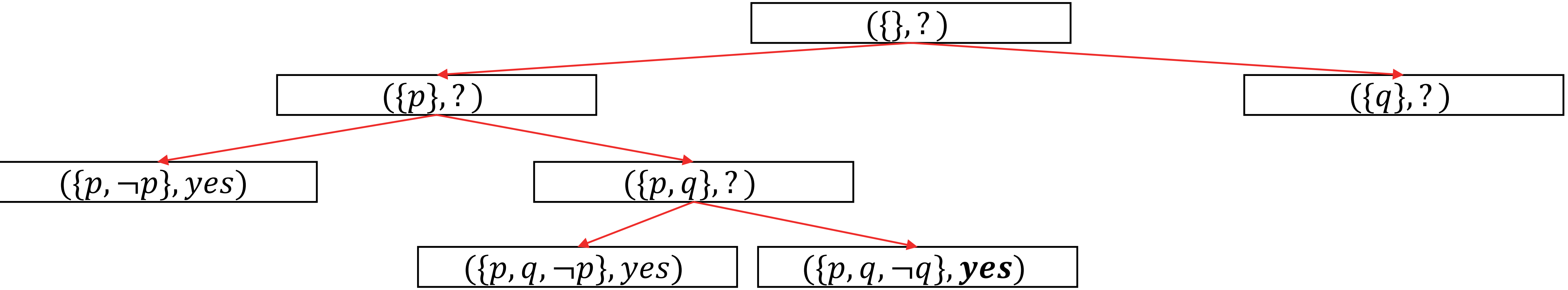
$p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q$



Modelelimination (IV)

- Solve the following problem instances:

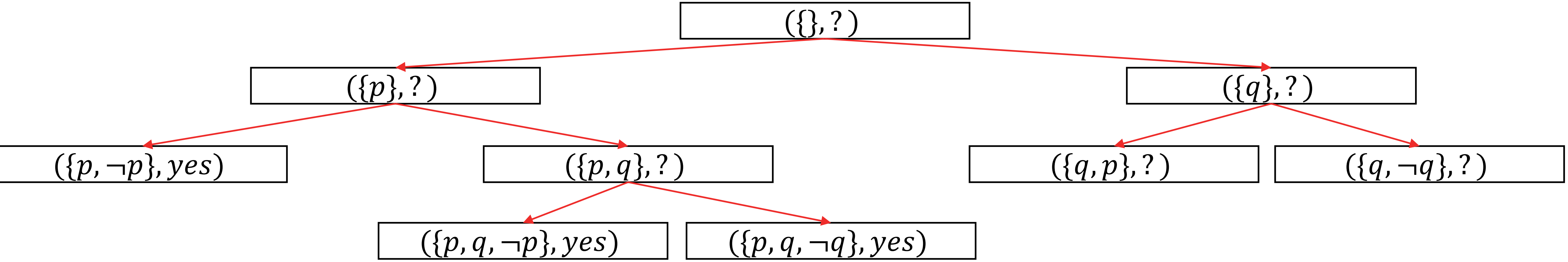
$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$



Modelelimination (IV)

- Solve the following problem instances:

$p \lor q, \boldsymbol{p} \lor \neg \boldsymbol{q}, \neg p \lor q, \neg p \lor \neg q$

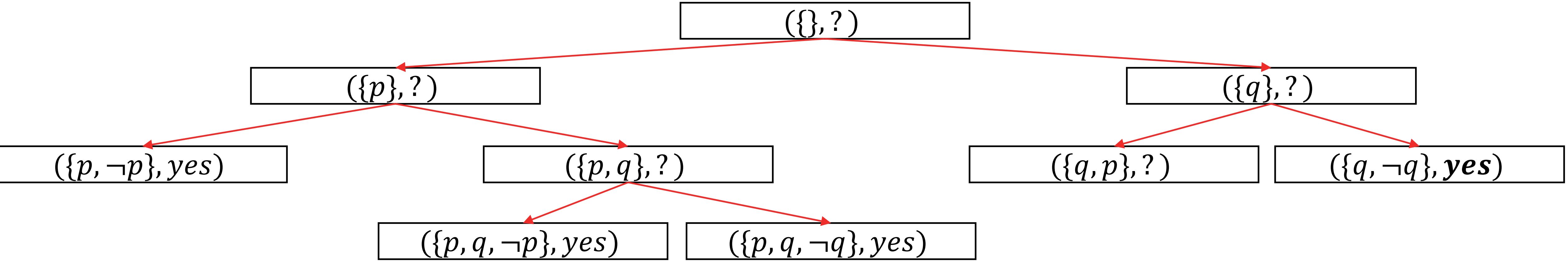


UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

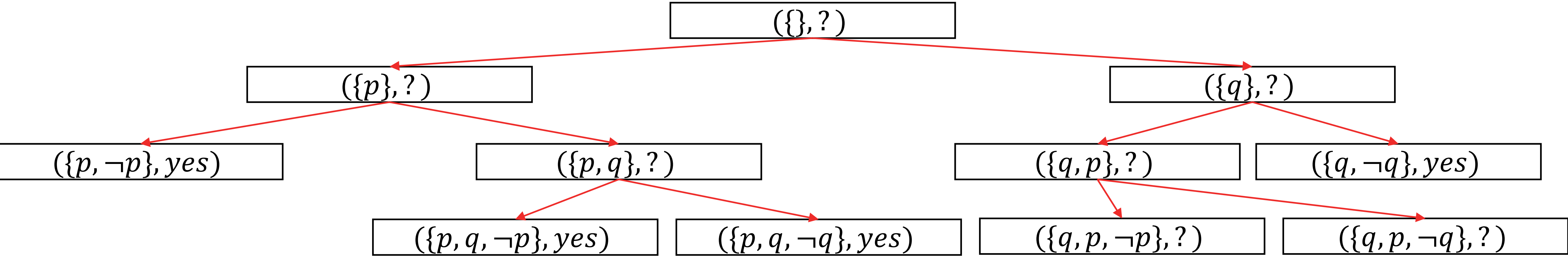
$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$



Modelelimination (IV)

- Solve the following problem instances:

$p \lor q, p \lor \neg q, \neg p \lor q, \boldsymbol{\neg p} \lor \boldsymbol{\neg q}$

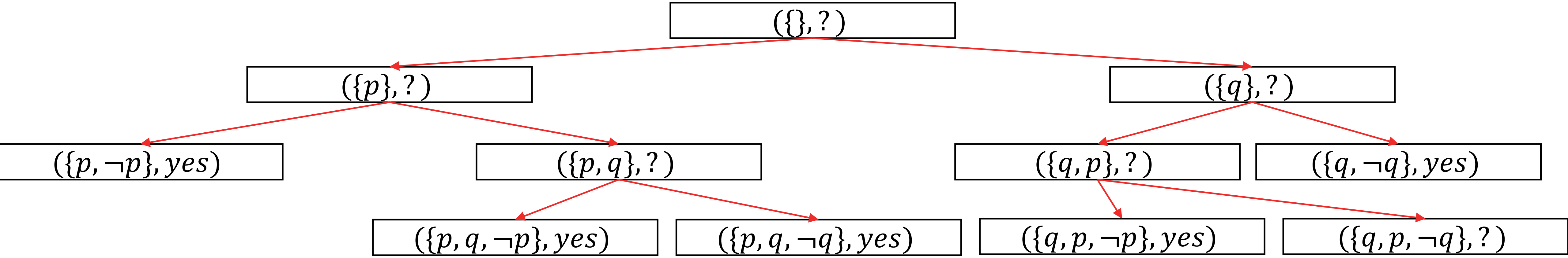


UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

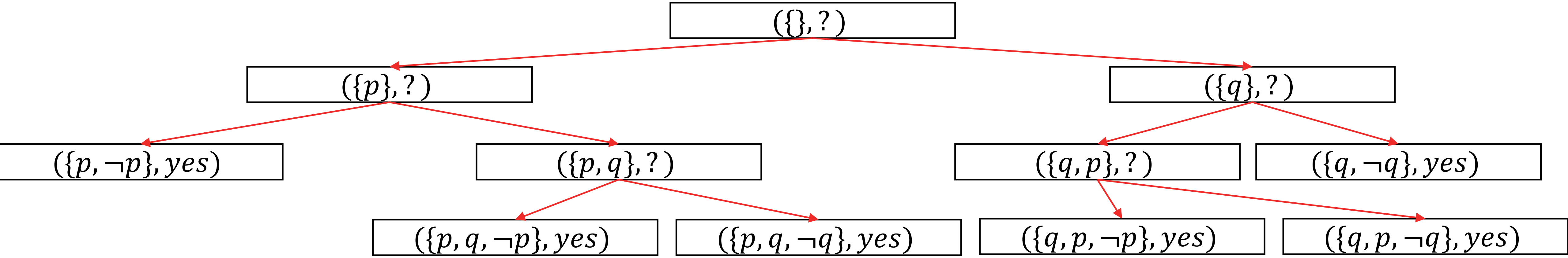
$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$



Modelelimination (IV)

- Solve the following problem instances:

$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

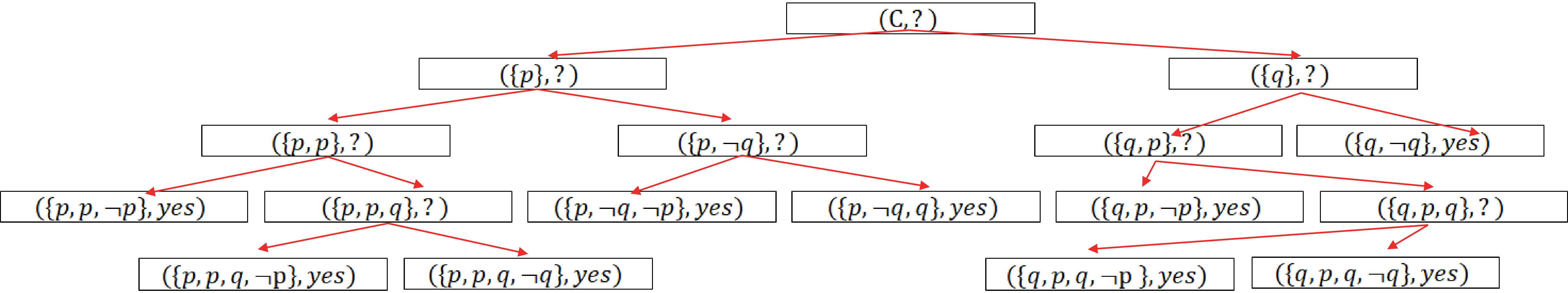


Modelelimination (IV)

- More than one way to solve! Search control matters!

$p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$

$\emptyset = C$



Modelelimination (IV)

- Solve the following problem instances:

$p\ , q, \neg\ q$
$\emptyset = C$

Modelelimination (IV)

- Again more than one way to solve:

$p\ ,q, \neg\ q$
$\emptyset = C$

$$(C, ?) \rightarrow (\{p\}, ?) \rightarrow (\{p, q\}, ?) \rightarrow (\{p, q, \neg q\}, yes)$$

$$(C, ?) \rightarrow (\{q\}, ?) \rightarrow (\{q, \neg q\}, yes)$$

Modelelimination (IV)

- Solve the following problem instances:

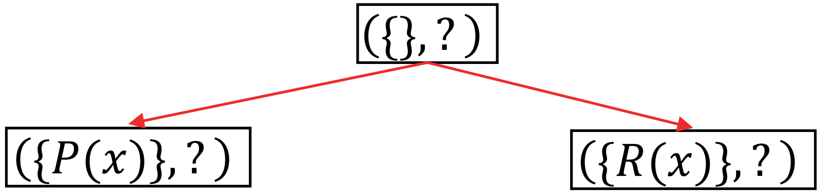
$P(x) \vee R(x)$, $\neg R(f(a,b))$, $\neg P(g(a,b))$

$\boxed{(\{\}, ?)}$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))



Modelelimination (IV)

- Solve the following problem instances:

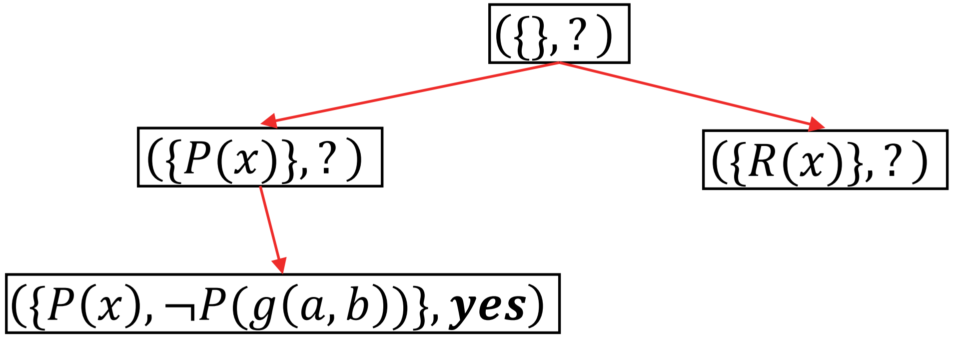
$P(x) \vee R(x)$, $\neg R(f(a,b))$, $\mathbf{\neg P(g(a,b))}$

$$\begin{tikzcd}
 & \boxed{(\{\},?)} \arrow[dl] \arrow[dr] & \\
\boxed{(\{P(x)\},?)} \arrow[d] & & \boxed{(\{R(x)\},?)} \\
\boxed{(\{P(x),\neg P(g(a,b))\},?)} & &
\end{tikzcd}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))



$$\boldsymbol{mgu = \{x \approx g(a,b)\}}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), **¬R(f(a,b)),** ¬P(g(a,b))

$$\boxed{(\{\}, ?)}$$

$$\boxed{(\{P(x)\}, ?)} \qquad \boxed{(\{R(x)\}, ?)}$$

$$\boxed{(\{P(x), \neg P(g(a,b))\}, yes)} \qquad \boxed{(\{R(x), \neg R(f(a,b))\}, ?)}$$

$$mgu = \{x \approx g(a,b)\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$
\begin{tikzcd}
 & \boxed{(\{\},?)} \arrow[dl] \arrow[dr] & \\
\boxed{(\{P(x)\},?)} \arrow[d] & & \boxed{(\{R(x)\},?)} \arrow[d] \\
\boxed{(\{P(x),\neg P(g(a,b))\},yes)} & & \boxed{(\{R(x),\neg R(f(a,b))\},?)}
\end{tikzcd}
$$

$$mgu = \{x \approx g(a,b), \boldsymbol{x} \approx \boldsymbol{f}(\boldsymbol{a},\boldsymbol{b})\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) $\vee$ R(x), $\neg$R(f(a,b)), $\neg$P(g(a,b))

$(\{\}, ?)$

$(\{P(x)\}, ?)$ $(\{R(x)\}, ?)$

$(\{P(x), \neg P(g(a,b))\}, yes)$ $(\{R(x), \neg R(f(a,b))\}, ?)$

backtrack

$mgu = \{x \approx g(a,b), x \approx f(a,b)\}$

UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$\begin{tikzcd}[column sep=small]
& \boxed{(\{\},?)} \arrow[dl] \arrow[dr] & \\
\boxed{(\{P(x)\},?)} \arrow[d] & & \boxed{(\{R(x)\},?)} \\
\boxed{(\{P(x),\neg P(g(a,b))\},yes)} & &
\end{tikzcd}$$

backtrack

$$mgu = \{x \approx g(a,b)\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) $\vee$ R(x), $\neg$R(f(a,b)), $\neg$P(g(a,b))

$\boxed{(\{\}, ?)}$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), **¬R(f(a,b)),** ¬P(g(a,b))

$$\boxed{(\{\},?)}$$
$$\downarrow$$
$$\boxed{(\{\neg R(f(a,b))\},?)}$$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$\boxed{(\{\}, ?)}$$

$$\boxed{(\{\neg R(f(a,b))\}, ?)}$$

$$\boxed{\left(\{\neg R\left(f(a,b)\right), P(x)\}, ?\right)} \qquad \boxed{\left(\{\neg R\left(f(a,b)\right), R(x)\}, ?\right)}$$

$$mgu = \{\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$(\{\}, ?)$$

$$(\{\neg R(f(a,b))\}, ?)$$

$$(\{\neg R(f(a,b)), P(x)\}, ?) \qquad (\{\neg R(f(a,b)), R(x)\}, \boldsymbol{yes})$$

$$\boldsymbol{mgu} = \{\boldsymbol{x} \approx \boldsymbol{f}(\boldsymbol{a}, \boldsymbol{b})\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), **¬P(g(a,b))**

$$\begin{tikzcd}
& \boxed{(\{\},?)} \arrow[d] & \\
& \boxed{(\{\neg R(f(a,b))\},?)} \arrow[dl] \arrow[dr] & \\
\boxed{(\{\neg R(f(a,b)),P(x)\},?)} \arrow[d] & & \boxed{(\{\neg R(f(a,b)),R(x)\},yes)} \\
\boxed{(\{\neg R(f(a,b)),P(x),\neg P(g(a,b))\},?)} & &
\end{tikzcd}$$

$$mgu = \{x \approx f(a,b)\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) $\vee$ R(x), $\neg$R(f(a,b)), $\neg$P(g(a,b))

$$
\begin{tikzcd}
& \boxed{(\{\}, ?)} \arrow[d] & \\
& \boxed{(\{\neg R(f(a,b))\}, ?)} \arrow[dl] \arrow[dr] & \\
\boxed{(\{\neg R(f(a,b)), P(x)\}, ?)} \arrow[d] & & \boxed{(\{\neg R(f(a,b)), R(x)\}, yes)} \\
\boxed{(\{\neg R(f(a,b)), P(x), \neg P(g(a,b))\}, ?)} & \textit{backtrack} &
\end{tikzcd}
$$

$$mgu = \{x \approx f(a,b), x \approx g(a,b)\}$$

UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

$P(x) \vee R(x)$, $\neg R(f(a,b))$, $\neg P(g(a,b))$

$$\boxed{(\{\},?)}$$

$$\boxed{(\{\neg R(f(a,b))\},?)}$$

backtrack

$$\boxed{(\{\neg R(f(a,b)), P(x)\},?)} \qquad \boxed{(\{\neg R(f(a,b)), R(x)\}, yes)}$$

$$mgu = \{x \approx f(a,b))\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$\boxed{(\{\}, ?)}$

backtrack

$\downarrow$

$\boxed{(\{\neg R(f(a,b))\}, ?)}$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

$P(x) \vee R(x)$, $\neg R(f(a,b))$, $\neg P(g(a,b))$

$\boxed{(\{\}, ?)}$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), **¬P(g(a,b))**

$$\begin{tikzcd} \boxed{(\{\}, ?)} \arrow[d] \\ \boxed{(\{\neg P(g(a,b))\}, ?)} \end{tikzcd}$$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$\boxed{(\{\},?)}$$

$$\boxed{(\{\neg P(g(a,b))\},?)}$$

$$\boxed{(\{\neg P(g(a,b)), P(x)\},?)} \qquad \boxed{(\{\neg P(g(a,b)), R(x)\},?)}$$

$$mgu = \{\}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$
\begin{array}{ccc}
 & \boxed{(\{\}, ?)} & \\
 & \downarrow & \\
 & \boxed{(\{\neg P(g(a,b))\}, ?)} & \\
\swarrow & & \searrow \\
\boxed{(\{\neg P(g(a,b)), P(x)\}, \boldsymbol{yes})} & & \boxed{(\{\neg P(g(a,b)), R(x)\}, ?)}
\end{array}
$$

$$\boldsymbol{mgu = \{x \approx g(a,b)\}}$$

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), **¬R(f(a,b)),** ¬P(g(a,b))

$$\boxed{(\{\}, ?)}$$

$$\boxed{(\{\neg P(g(a,b))\}, ?)}$$

$$\boxed{(\{\neg P(g(a,b)), P(x)\}, yes)} \qquad \boxed{(\{\neg P(g(a,b)), R(x)\}, yes)}$$

$$\boxed{(\{\neg P(g(a,b)), R(x), \neg R(f(a,b))\}, ?)}$$

$$mgu = \{x \approx g(a,b)\}$$

UNIVERSITY OF CALGARY

Modelelimination (IV)

- Solve the following problem instances:

P(x) ∨ R(x), ¬R(f(a,b)), ¬P(g(a,b))

$$\boxed{(\{\},?)}$$

$$\boxed{(\{\neg P(g(a,b))\},?)}$$

$$\boxed{(\{\neg P(g(a,b)),P(x)\},yes)} \qquad \boxed{(\{\neg P(g(a,b)),R(x)\},yes)}$$

$$\boxed{(\{\neg P(g(a,b)),R(x),\neg R(f(a,b))\},?)}$$

backtrack

$$mgu = \{x \approx g(a,b), x \approx f(a,b)\}$$

Modelelimination (IV)

- Solve the following problem instances:

$P(x) \vee R(x)$, $\neg R(f(a,b))$, $\neg P(g(a,b))$

$\boxed{(\{\}, ?)}$

$mgu = \{\}$

Modelelimination (IV)

- Solve the following problem instances:

1) $p \lor q, p \lor \neg q, \neg p \lor q, \neg p \lor \neg q$ success
2) $p, q, \neg q$ success
3) $P(x) \lor R(x), \neg R\big(f(a,b)\big), \neg P(g(a,b))$ failure

Model-elimination: Tree-Based

Model-elimination (IV)

Tasks:

- Describe Model-elimination as and-tree-based search model
- Describe formally a search control for your model that uses backtracking to avoid generating an infinite branch in the tree representing the state (if the problem instance is solvable)

Model-elimination (IV)

Describe Model-elimination as and-tree-based search model

- We have set of Clauses $C = \{c_1, \ldots, c_p\}$ of p clauses where is clause $c_i \in C$ is of form $c_i = L_1 \vee \cdots \vee L_n$(disjunction of literals) so will define a set all literals $L_{all} = \{L_j \mid L_j \text{ from } c_i \ \forall c_i \in C\}$ (set of all literals present in C)
- $Prob = \{pr_1, \ldots, pr_m\}$ where a $pr_i \in Prob$ is
  - $pr_i \in 2^{L_{all}}$
  - (a single problem is some subset of L_j parts or $Prob = 2^{L_{all}}$)
- Div will be defined by the relationship that if $pr \in Prob$ is selected to divide into sub-problems then based a choice of $c_i \in C$ where $c_i = L_1 \vee \cdots \vee L_n$ then n sub-problems are created where each sub-problem pr_j fulfills
  - $pr_j = pr \ \cup L_j$
  - (each sub-problem j is a combination of the existing set of literals with the j^{th} literal)

Model-elimination (IV)

Describe Model-elimination as and-tree-based search model

- Div will be defined by the relationship that if $pr \in Prob$ is selected to divide into sub-problems then based a choice of $c_i \in C$ where $c_i = L_1 \vee \cdots \vee L_n$ then n sub-problems are created where each sub-problem pr_j fulfills
  - $pr_j = pr \cup L_j$
  - (each sub-problem j is a combination of the existing set of literals with the j^{th} literal)
  - If we want to avoid infinite divisions me might also add that one pr_j must be created such that the L_j being added is such that $\neg L_j \in pr$. We are eliminating one model sub-branch already (unless pr = {} at root)

UNIVERSITY OF CALGARY

Model-elimination (IV)

Describe formally a search control for your model

f_{leaf} =

1. 0 if (pr,?) contains P and ¬P′ such that there is a σ with σ(P) ≡ σ(P′) $(tie\ break\ by <_{Lit})$
2. $|pr|$ otherwise $(tie\ break\ by <_{Lit})$

f_{trans} =

1. (pr, yes) if (pr,?) contains P and ¬P′ such that there is a σ with σ(P) ≡ σ(P′)
2. if out of unique $c_i \in C$ for more Div or fail unfication then backtrack (and remove backtracked $c_j \in C$ from future consideration for Div at that leaf)
3. select $c_i \in C$ that has most negations (tie break by $<_{Lit}$) for Div

Remarks

- There are many optimization problems that can be solved by an **and-tree**-based search without backtracking!
- Backtracking is often used to reduce the memory needs for a search (it allows to store only one path of the tree).
- Backtracking can always be avoided by using **and-or-tree**-based search.
- Branch-and-bound, dynamic programming and a lot of other algorithm schemes are **and-tree**-based search! (Think about how standard code/functions work using a stack frame to store history!)

Onward to …
Or-Tree-based Search

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF
CALGARY