SAARLAND UNIVERSITY ARTIFICIAL INTELLIGENCE

Prof. Jana Koehler, Dr. Álvaro Torralba

Exercise Sheet 5.

Solution

Exercise 18: Unification Algorithm. (2 Points)

Run the unification algorithm from the lecture on each of the given sets. Assume v, w, x, y, z to be *variables*, and a, b, c to be *constants*. **For each step, write down the set to unify, the disagreement set, and the updated set of substitutions, i.e.,** $T_0, D_0, s_1, T_1, D_1, \ldots$ **until the algorithm terminates** (cf. Chapter 8, slide 26-27). In particular, if the algorithm does not output a failure, list the *entire* content of the substitutions $s_1, \ldots, s_n$.

(i) $\{P(x, f(a, y), g(x)), P(x, f(a, b), z)\}$

(ii) $\{P(f(x, a), y), P(f(b, g(z)), g(c))\}$

(iii) $\{P(z, x), P(f(x), z)\}$

(iv) $\{P(x, f(b, y, b)), P(a, v), P(w, f(z, w, b))\}$

(Solution)

(i) $\{P(x, f(a, y), g(x)), P(x, f(a, b), z)\}$

$$\begin{aligned} D_0 &= \{y, b\} \\ s_1 &= \left\{\frac{y}{b}\right\} \\ T_1 &= \{P(x, f(a, b), g(x)), P(x, f(a, b), z)\} \end{aligned}$$

$$\begin{aligned} D_1 &= \{g(x), z\} \\ s_2 &= s_1\left\{\frac{z}{g(x)}\right\} = \left\{\frac{y}{b}, \frac{z}{g(x)}\right\} \\ T_2 &= \{P(x, f(a, b), g(x)), P(x, f(a, b), g(x))\} = \{P(x, f(a, b), g(x))\} \end{aligned}$$

T_2 is a singleton, so stop and output $s_2 = \{\frac{y}{b}, \frac{z}{g(x)}\}$.

(ii) $\{P(f(x,a),y), P(f(b,g(z)),g(c))\}$

$$\begin{aligned} D_0 &= \{x, b\} \\ s_1 &= \left\{\frac{x}{b}\right\} \\ T_1 &= \{P(f(b,a),y), P(f(b,g(z)),g(c))\} \\ \\ D_1 &= \{a, g(z)\} \end{aligned}$$

No variable is contained in D_1. Stop and output "failure".

(iii) $\{P(z,x), P(f(x),z)\}$

$$\begin{aligned} D_0 &= \{z, f(x)\} \\ s_1 &= \left\{\frac{z}{f(x)}\right\} \\ T_1 &= \{P((f(x),x), P(f(x),f(x))\} \\ \\ D_1 &= \{x, f(x)\} \end{aligned}$$

There is no pair of a term t and a variable x such that x is not contained in t. Stop and output "failure"

(iv) $\{P(x, f(b, y, b)), P(a, v), P(w, f(z, w, b))\}$

$$\begin{aligned}
D_0 &= \{x, a, w\} \\
s_1 &= \left\{\frac{x}{a}\right\} \\
T_1 &= \{P(a, f(b, y, b)), P(a, v), P(w, f(z, w, b))\} \\
D_1 &= \{a, w\} \\
s_2 &= s_1\left\{\frac{w}{a}\right\} = \left\{\frac{x}{a}, \frac{w}{a}\right\} \\
T_2 &= \{P(a, f(b, y, b)), P(a, v), P(a, f(z, a, b))\} \\
D_2 &= \{f(b, y, b), v, f(z, a, b)\} \\
s_3 &= s_2\left\{\frac{v}{f(b, y, b)}\right\} = \left\{\frac{x}{a}, \frac{w}{a}, \frac{v}{f(b, y, b)}\right\} \\
T_3 &= \{P(a, f(b, y, b)), P(a, f(b, y, b)), P(a, f(z, a, b))\} = \{P(a, f(b, y, b)), P(a, f(z, a, b))\} \\
D_3 &= \{b, z\} \\
s_4 &= s_3\left\{\frac{z}{b}\right\} = \left\{\frac{x}{a}, \frac{w}{a}, \frac{v}{f(b, y, b)}, \frac{z}{b}\right\} \\
T_4 &= \{P(a, f(b, y, b)), P(a, f(b, a, b))\} \\
D_4 &= \{y, a\} \\
s_5 &= s_4\left\{\frac{y}{a}\right\} = \left\{\frac{x}{a}, \frac{w}{a}, \frac{v}{f(b, a, b)}, \frac{z}{b}, \frac{y}{a}\right\} \\
T_5 &= \{P(a, f(b, a, b)), P(a, f(b, a, b))\} = \{P(a, f(b, a, b))\}
\end{aligned}$$

T_5 is a singleton, so stop and output $s_5 = \left\{\frac{x}{a}, \frac{w}{a}, \frac{v}{f(b,a,b)}, \frac{z}{b}, \frac{y}{a}\right\}$.

(/Solution)

Exercise 19: PL1 resolution. (3 Points)

Consider now the following set of predicate logic formulas in Skolem normal form:

1. Every student is cool.
   $\varphi_1 = \forall x[\neg Student(x) \vee Cool(x)]$

2. Olaf or Sven are students.
   $\varphi_2 = Student(Olaf) \vee Student(Sven)$

3. All people that are cool watch a television series. Note: $f(x)$ comes from removing an existential quantification ($\exists y[Series(y) \wedge Watches(x, y)]$) and represents the series

x watches.
$\varphi_3 = \forall x[\neg Cool(x) \vee (Series(f(x)) \wedge Watches(x, f(x)))]$

4. Everyone who is cool owns a cat.
$\varphi_4 = \forall x[\neg Cool(x) \vee Owns(x, cat)]$

5. Olaf does not watch any television series.
$\varphi_5 = \forall x[\neg Series(x) \vee \neg Watches(Olaf, x)]$

We want to prove by contradiction that Sven owns a cat. Therefore, we add to our set of clauses:

6. Sven does not own a cat.
$\varphi_6 = \neg Owns(Sven, cat)$

Perform the following operations:

(i) Write down the **set of clauses Δ corresponding to the clausal normal forms** of these formulas.

(ii) Use **binary PL1 resolution** to show that Δ is unsatisfiable.
Note: If you need to use a clause more than once, make a copy of the clause and rename the variables to avoid name collisions.

(Solution)

(i) The clauses for each formula are:

$\varphi_1 = \forall x[\neg Student(x) \vee Cool(x)]$
$\varphi_2 = Student(Olaf) \vee Student(Sven)$
$\varphi_3 = \forall x[\neg Cool(x) \vee (Series(f(x)) \wedge Watches(x, f(x)))]$
$\varphi_4 = \forall x[\neg Cool(x) \vee Owns(x, cat)]$
$\varphi_5 = \forall x[\neg Series(x) \vee \neg Watches(Olaf, x)]$
$\varphi_6 = \neg Owns(Sven, cat)$

The only formula that is not in CNF is φ_3. This can be transformed into:
$\varphi_3 = \{\neg Cool(x), Series(f(x))\}, \{\neg Cool(x), Watches(x, f(x))\}$

$\varphi_1 = \{\neg Student(x), Cool(x)\}$
$\varphi_2 = \{Student(Olaf), Student(Sven)\}$
$\varphi_3 = \{\neg Cool(x), Series(f(x))\}, \{\neg Cool(x), Watches(x, f(x))\}$
$\varphi_4 = \{\neg Cool(x), Owns(x, cat)\}$
$\varphi_5 = \{\neg Series(x), \neg Watches(Olaf, x)\}$
$\varphi_6 = \{\neg Owns(Sven, cat)\}$

When creating Δ, do not forget to standardize the variables apart!

$\varphi_1 = \{\neg Student(x), Cool(x)\}$
$\varphi_2 = \{Student(Olaf), Student(Sven)\}$
$\varphi_3 = \{\neg Cool(y_1), Series(f(y_1))\}, \{\neg Cool(y_2), Watches(y_2, f(y_2))\}$
$\varphi_4 = \{\neg Cool(z), Owns(z, cat)\}$
$\varphi_5 = \{\neg Series(t), \neg Watches(Olaf, t)\}$
$\varphi_6 = \{\neg Owns(Sven, cat)\}$

(ii) Figure 1 shows a suitable resolution tree.

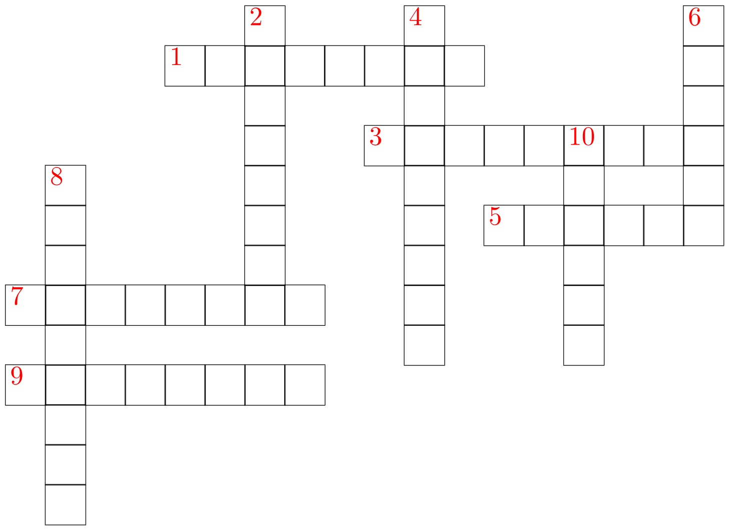
(/Solution)

$$\begin{tikzcd}
\psi_4 \arrow[dr, "\frac{x}{Sven}"] & & \psi_6 \arrow[dl] & & \\
& \{\neg Cool(Sven)\} \arrow[dr] & & \psi_1' = \{\neg Student(x'), Cool(x')\} \arrow[dl, "\frac{x'}{Sven}"] & \\
\psi_2 \arrow[dr] & & \{\neg Student(Sven))\} \arrow[dl] & & \\
& \{Student(Olaf)\} \arrow[dl] & & \psi_1'' = \{\neg Student(x''), Cool(x'')\} \arrow[lll, dl, "\frac{x''}{Olaf}"] & \\
\{Cool(Olaf)\} \arrow[drr] \arrow[ddddrrr, bend right] & & & & \{\neg Cool(y_1), Series(f(y_1))\} \arrow[dll, "\frac{y_1}{Olaf}"] \\
& & \{Series(f(Olaf))\} \arrow[d] & & \psi_5 \arrow[dll, "\frac{t}{f(Olaf)}"] \\
& & \{\neg Watches(Olaf, f(Olaf))\} \arrow[dr] & & \{\neg Cool(y_2), Watches(y_2, f(y_2))\} \arrow[dl, "\frac{y_2}{Olaf}"] \\
& & & \{\neg Cool(Olaf)\} \arrow[d] & \\
& & & \square &
\end{tikzcd}$$

Figure 1: Solution of Exercise 19 (b)

Exercise 20: Formulating a constraint network. (2 Points)

Consider the crossword puzzle in Figure 2. Even numbers stand for words to be inserted vertically, odd numbers stand for words to be inserted horizontally. Each word can be added only once.



`algorithm`	`inference`
`DeepBlue`	`DeepMind`
`Libratus`	`search`
`computer`	`spaces`
`heuristic`	`states`

Figure 2: Crossword puzzle to be formalized in Exercise 20.

Formulate this puzzle as a **constraint network whose solutions are the solutions to the puzzle.** Use the set of variables $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$, where variable v_i models which words can be put into gap i. The domain of each variable is a subset of the words to be inserted into the puzzle.

1. Specify the domain of each variable, i.e., give D_{v_i}, for each variable $v_i \in V$.

2. Specify the constraints, i.e., give $C_{v_i v_j} \subseteq D_{v_i} \times D_{v_j}$, for all relevant pairs of variables. Do not include the pairs where the set of feasible values is equal to the entire set $D_{v_i} \times D_{v_j}$.

(Solution)

1. The domain of each variable, is the subset of words that have the correct length:

$$\begin{aligned} D_{v_1} = D_{v_7} = D_{v_9} = D_{v_2} &= \{\texttt{DeepBlue}, \texttt{DeepMind}, \texttt{computer}, \texttt{Libratus}\} \\ D_{v_3} = D_{v_4} = D_{v_8} &= \{\texttt{algorithm}, \texttt{heuristic}, \texttt{inference}\} \\ D_{v_5} = D_{v_6} = D_{v_{10}} &= \{\texttt{search}, \texttt{spaces}, \texttt{states}\} \end{aligned}$$

2. There are two kinds of non-trivial constraints, i.e., feasible value pairs not equal to the entire set $D_{v_i} \times D_{v_j}$.

   One kind of constraint prevents the same word from being used twice:

$$C_{v_i v_j} = \big\{(d, d') \mid d \in D_{v_i}, d' \in D_{v_j}, d \neq d'\big\}$$

   for all non-intersecting pairs $v_i, v_j \in V$ such that $D_{v_i} = D_{v_j}, i \neq j$.

   The other kind of constraint involves the pairs of variables whose placeholders intersect in a way that invalidates some combinations:

$$\begin{aligned} C_{v_1 v_2} &= \big\{(\texttt{de}\underline{\texttt{e}}\texttt{pmind}, \texttt{d}\underline{\texttt{e}}\texttt{epblue}), (\texttt{d}\underline{\texttt{e}}\texttt{epblue}, \texttt{de}\underline{\texttt{e}}\texttt{pmind})\big\} \\ C_{v_1 v_4} &= \big\{(\texttt{deepmi}\underline{\texttt{n}}\texttt{d}, \texttt{i}\underline{\texttt{n}}\texttt{ference}), (\texttt{comput}\underline{\texttt{e}}\texttt{r}, \texttt{h}\underline{\texttt{e}}\texttt{uristic})\big\} \\ C_{v_2 v_7} &= \big\{(\texttt{deepblu}\underline{\texttt{e}}, \texttt{comput}\underline{\texttt{e}}\texttt{r})\big\} \\ C_{v_3 v_4} &= \big\{(\texttt{h}\underline{\texttt{e}}\texttt{uristic}, \texttt{inf}\underline{\texttt{e}}\texttt{rence})\big\} \\ C_{v_3 v_6} &= \big\{(\texttt{heuristi}\underline{\texttt{c}}, \texttt{spa}\underline{\texttt{c}}\texttt{es})\big\} \\ C_{v_3 v_{10}} &= \big\{(\texttt{heuri}\underline{\texttt{s}}\texttt{tic}, d) \mid d \in D_{v_{10}}\big\} \\ C_{v_5 v_6} &= \big\{(\texttt{space}\underline{\texttt{s}}, \texttt{state}\underline{\texttt{s}}), (\texttt{state}\underline{\texttt{s}}, \texttt{space}\underline{\texttt{s}})\big\} \\ C_{v_5 v_{10}} &= \big\{(d, d') \mid d \in D_{v_5}, d' \in D_{v_{10}}, d \neq d'\big\} \\ C_{v_7 v_8} &= \big\{(\texttt{c}\underline{\texttt{o}}\texttt{mputer}, \texttt{alg}\underline{\texttt{o}}\texttt{rithm}), (\texttt{d}\underline{\texttt{e}}\texttt{epblue}, \texttt{inf}\underline{\texttt{e}}\texttt{rence}), (\texttt{d}\underline{\texttt{e}}\texttt{epmind}, \texttt{inf}\underline{\texttt{e}}\texttt{rence})\big\} \\ C_{v_8 v_9} &= \big\{(\texttt{algor}\underline{\texttt{i}}\texttt{thm}, \texttt{l}\underline{\texttt{i}}\texttt{bratus}), (\texttt{infer}\underline{\texttt{e}}\texttt{nce}, \texttt{d}\underline{\texttt{e}}\texttt{epblue}), (\texttt{infer}\underline{\texttt{e}}\texttt{nce}, \texttt{d}\underline{\texttt{e}}\texttt{epmind})\big\} \end{aligned}$$

(/Solution)

Exercise 21: Naïve backtracking algorithm. (3 Points)

Given the map coloring problem illustrated in Figure 3, run the naïve backtracking algorithm to find a coloring such that all adjacent regions are painted in a different color. The corresponding constraint satisfaction problem $\gamma = (V, D, C)$ has variables $V = \{$Alsace, Champagne, Lorraine, Luxembourg, Rhineland-Palatinate, Saarland, Wallonia$\}$ for each of the regions, each with domain $D = \{\text{red}, \text{green}, \text{blue}, \text{yellow}\}$. The constraints are such that no pair of adjacent regions can have the same color, i.e., $C_{\{u,v\}} \in C$ for all adjacent u and v, $u \neq v$. For example, $C_{\{\text{Saarland},\text{Lorraine}\}} \in C$ because Saarland and Lorraine are adjacent and consequently require two different colors, but $C_{\{\text{Saarland},\text{Alsace}\}} \notin C$.

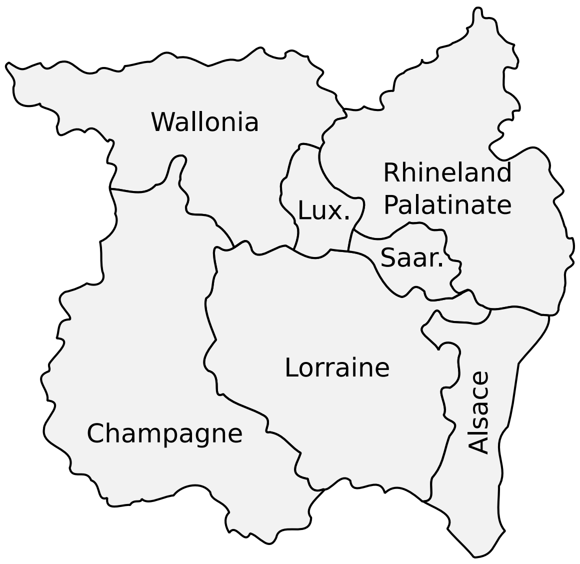


Figure 3: A map of Saarland and surrounding regions.

1. Run the naïve backtracking algorithm using the combination strategy described on slide 42 (bottom) of Chapter 9, i.e., **among the most *constrained* variables, pick the most *constraining* first. As a tie-breaker, use the alphabetical order on the region labels**. The value order should first assign red, then green, then blue, and eventually yellow. **Draw the search graph and mark inconsistent assignments.**

2. When using the same value ordering (red → green → blue → yellow), define a variable ordering such that the search graph contains more nodes than the one in the previous part. Explain why more nodes are generated.

(Solution)

1. See Figure 4.
2. There are many orders that will result in a larger search graph. One example is, among the least constrained variables, to pick the least constraining first. This ordering does not detect constraint violations immediately since it assigns potentially invalid colors to non-bordering regions first. This leads to more nodes being expanded before a violation is encountered.

(/Solution)

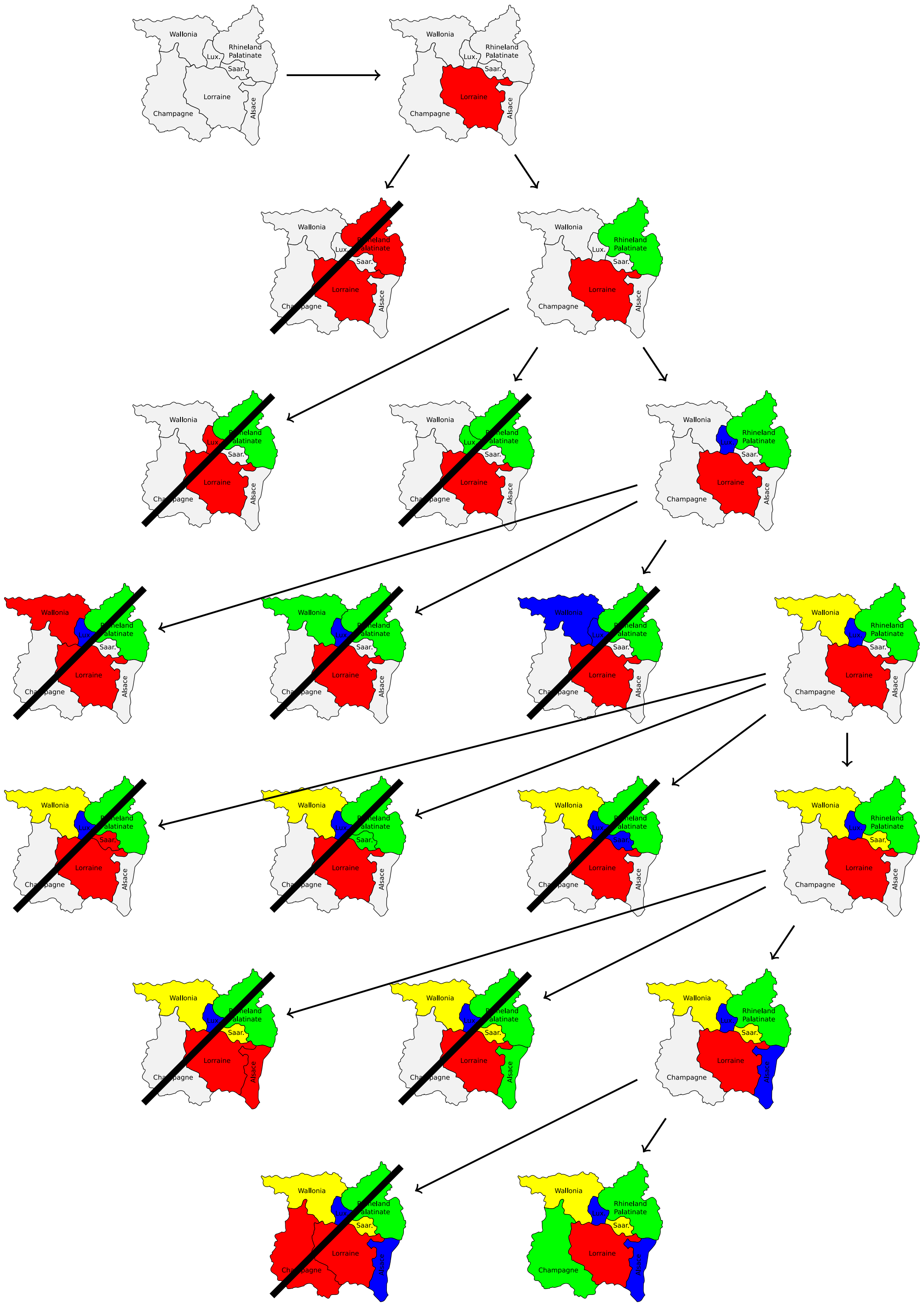


Figure 4: Solution of Exercise 21(a).