

# Abstract Interpretation

1. Using the “ground” abstraction of terms, abstract, the following system of equations and perform abstract unification:

$$\begin{aligned} X &= f(a,U), \\ Y &= f(V,b), \\ X &= Y, \\ g(U,W) &= g(V,h(Z)). \end{aligned}$$

Concrete equation system:  $\{X=f(a,U), Y=f(V,b), X=Y, g(U,W)=g(V,h(Z))\}$

Concrete unification:

- |   |                       |
|---|-----------------------|
| 1. $\{X=f(a,U), Y=f(V,b), X=Y, g(U,W)=g(V,h(Z))\}$      | peel                  |
| 2. $\{X=f(a,U), Y=f(V,b), X=Y, U=V, W=h(Z)\}$           | substitute $X=Y$      |
| 3. $\{Y=f(a,U), Y=f(V,b), X=Y, U=V, W=h(Z)\}$           | substitute $Y=f(a,U)$ |
| 4. $\{Y=f(a,U), f(a,U)=f(V,b), X=f(a,U), U=V, W=h(Z)\}$ | peel second           |
| 5. $\{Y=f(a,U), a=V, U=b, X=f(a,U), U=V, W=h(Z)\}$      | switch second         |
| 6. $\{Y=f(a,U), V=a, U=b, X=f(a,U), U=V, W=h(Z)\}$      | substitute $U=b$      |
| 7. $\{Y=f(a,b), V=a, U=b, X=f(a,b), b=V, W=h(Z)\}$      | switch $b=V$          |
| 8. $\{Y=f(a,b), V=a, U=b, X=f(a,b), V=b, W=h(Z)\}$      | substitute $V=a$      |
| 9. $\{Y=f(a,b), V=a, U=b, X=f(a,b), a=b, W=h(Z)\}$      | no solution           |

Abstrating the system yields the Abstract equation system:  $\{X=f(\mathcal{G},U), Y=f(V,\mathcal{G}), X=Y, g(U,W)=g(V,h(Z)), \mathcal{G}=\mathcal{G}\}$

Abstract unification:

- |  |                                 |
|--|---------------------------------|
| 1. $\{X=f(\mathcal{G},U), Y=f(V,\mathcal{G}), X=Y, g(U,W)=g(V,h(Z)), \mathcal{G}=\mathcal{G}\}$                          | peel                            |
| 2. $\{X=f(\mathcal{G},U), Y=f(V,\mathcal{G}), X=Y, U=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$                               | substitute $X=Y$                |
| 3. $\{Y=f(\mathcal{G},U), Y=f(V,\mathcal{G}), X=Y, U=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$                               | substitute $Y=f(\mathcal{G},U)$ |
| 4. $\{Y=f(\mathcal{G},U), f(\mathcal{G},U)=f(V,\mathcal{G}), X=f(\mathcal{G},U), U=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$ | peel second                     |
| 5. $\{Y=f(\mathcal{G},U), \mathcal{G}=V, U=\mathcal{G}, X=f(\mathcal{G},U), U=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$      | switch second                   |
| 6. $\{Y=f(\mathcal{G},U), V=\mathcal{G}, U=\mathcal{G}, X=f(\mathcal{G},U), U=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$      | substitute $U=\mathcal{G}$      |
| 7. $\{Y=\mathcal{G}, V=\mathcal{G}, U=\mathcal{G}, X=\mathcal{G}, \mathcal{G}=V, W=h(Z), \mathcal{G}=\mathcal{G}\}$      | switch $\mathcal{G}=V$          |
| 8. $\{Y=\mathcal{G}, V=\mathcal{G}, U=\mathcal{G}, X=\mathcal{G}, W=h(Z), \mathcal{G}=\mathcal{G}\}$                     | substitute $V=\mathcal{G}$      |
| 9.   | is solved form                  |

How close does it mimick concrete unification?

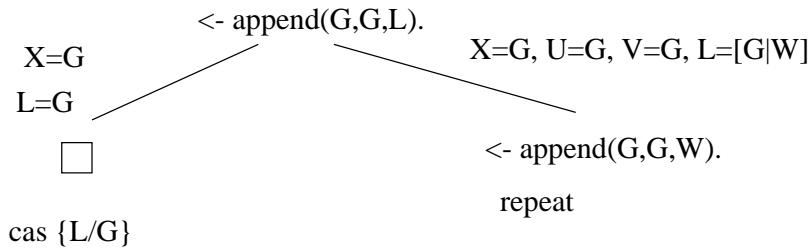
Close until step 8; in that step, the concrete equation produces  $a=b$  whereas the abstract step produces an equation  $\mathcal{G}=\mathcal{G}$  which is absorbed by the existing  $\mathcal{G}=\mathcal{G}$  equation.

2. Given the append program:

$append([], X, X).$

$append([X|U], V, [X|W]) \leftarrow append(U, V, W).$

construct an SLD tree for the abstract query  $\leftarrow append(\mathcal{G}, \mathcal{G}, L)$ , using abstract unification instead of concrete unification. Which abstract answers do you obtain?



The tree has an infinite branch with its nodes renamings of the initial goal. One can easily see that all answers are reduced to the form  $\{L/\mathcal{G}\}$  (e.g. the solved form of  $L = [\mathcal{G}|W]$ ,  $W = \mathcal{G}$  is  $L = \mathcal{G}$ ,  $W = \mathcal{G}$ ).

3. For the program

1.  $call\_app([a, b], [c], L)$ .
2.  $answer\_app([], X, X) \leftarrow call\_app([], X, X)$ .
3.  $call\_app(U, V, W) \leftarrow call\_app([X|U], V, [X|W])$ .
4.  $answer\_app([X|U], V, [X|W]) \leftarrow call\_app([X|U], V, [X|W]), answer\_app(U, V, W)$ .

(a) Compute the Least Herbrand Model

$$\begin{aligned}
T_P \uparrow 1 &= \{call\_app([a, b], [c], L) | L \in H_U\} \quad (1) \\
T_P \uparrow 2 &= T_P \uparrow 1 \cup \{call\_app([b], [c], L) | L \in H_U\} \quad (3) \\
T_P \uparrow 3 &= T_P \uparrow 2 \cup \{call\_app([], [c], L) | L \in H_U\} \quad (3) \\
T_P \uparrow 4 &= T_P \uparrow 3 \cup \{answer\_app([], [c], [c])\} \quad (2) \\
T_P \uparrow 5 &= T_P \uparrow 4 \cup \{answer\_app([b], [c], [b, c])\} \quad (4) \\
T_P \uparrow 6 &= T_P \uparrow 5 \cup \{answer\_app([a, b], [c], [a, b, c])\} \quad (4) \\
T_P \uparrow 7 &= TP \uparrow 6 \text{ fixpoint.}
\end{aligned}$$

Note that the computation almost simulates the top-down execution of SLD. A “non-ground”  $T_P$  operator working with equivalence classes of atoms that are variants of each other would give an exact simulation.

(b) Apply abstract compilation on it (using the  $\{g, \neg g\}$  domain)

1.  $call\_app(g, g, L)$ .
2.  $answer\_app(g, X, X) \leftarrow call\_app(g, X, X)$ .
3.  $call\_app(U, V, W) \leftarrow call\_app(L1, V, L3), j2(L1, X, U), j2(L3, X, W)$
4.  $answer\_app(L1, V, L3) \leftarrow call\_app(L1, V, L3), j2(L1, X, U), j2(L3, X, W), answer\_app(U, V, W)$ .

$$\begin{aligned}
j2(g, g, g) &\leftarrow \\
j2(\neg g, g, \neg g) &\leftarrow \\
j2(\neg g, \neg g, g) &\leftarrow \\
j2(\neg g, \neg g, \neg g) &\leftarrow
\end{aligned}$$

(c) Compute the Least Herbrand Model of the abstract program. What does it tell?

$$\begin{aligned}
T_P \uparrow 1 &= \{call\_app(g, g, g), call\_app(g, g, \neg g)\} \text{ or } \{call\_app(g, g, L) | L \in D\} \quad (1) \\
T_P \uparrow 2 &= T_P \uparrow 1 \cup \{answer\_app(g, g, g)\} \quad (2) \text{ (nothing new from clause (3))} \\
T_P \uparrow 3 &= T_P \uparrow 2 \text{ (nothing new from clause (4))}
\end{aligned}$$

One can see that all calls ( $call\_app/3$  atoms) have first two arguments ground and that all answers ( $answer\_app/3$  atoms) have all arguments ground.

4. Consider the program

$$\begin{aligned}
part(X, [], [], []) &\leftarrow . \\
part(X, [Y|L], [Y|L1], L2) &\leftarrow X \geq Y, part(X, L, L1, L2). \\
part(X, [Y|L], L1, [Y|L2]) &\leftarrow X < Y, part(X, L, L1, L2).
\end{aligned}$$

Using listlength as size, check that  $size(L) = size(L1) + size(L2)$  is a fixpoint for the predicate  $part(X, L, L1, L2)$  under the abstract  $T_P$  operator.

$$\begin{aligned}
T_P^{\alpha}(\{x_2 = x_3 + x_4\}) & \\
&= \alpha(T_P(\gamma(\{x_2 = x_3 + x_4\}))) \\
&= \alpha(T_P(\{part(X, L, L1, L2) | X, L, L1, L2 \in H_U \wedge size(L) = size(L1) + size(L2)\})) \\
&= \alpha(\{part(X, [], [], []) | X \in H_U\} \cup \\
&\quad \{part(X, [Y|L], [Y|L1], L2) | X, Y, L, L1, L2 \in H_U \wedge
\end{aligned}$$

$$\begin{aligned}
& \{X \geq Y \wedge \text{size}(L) = \text{size}(L1) + \text{size}(L2)\} \cup \\
& \{\text{part}(X, [Y|L], L1, [Y|L2]) | X, Y, L, L1, L2 \in H_U \wedge \\
& \quad X < Y \wedge \text{size}(L) = \text{size}(L1) + \text{size}(L2)\} \\
= & \text{lub}(\alpha(\{\text{part}(X, [], [], []) | X \in H_U\}), \\
& \alpha(\{\text{part}(X, [Y|L], [Y|L1], L2) | X, Y, L, L1, L2 \in H_U \wedge \\
& \quad X \geq Y \wedge \text{size}(L) = \text{size}(L1) + \text{size}(L2)\}), \\
& \alpha(\{\text{part}(X, [Y|L], L1, [Y|L2]) | X, Y, L, L1, L2 \in H_U \wedge \\
& \quad X < Y \wedge \text{size}(L) = \text{size}(L1) + \text{size}(L2)\})) \\
= & \text{lub}(\{x_2 = 0, x_3 = 0, x_4 = 0\}, \{x_2 = x_3 + x_4\}, \{x_2 = x_3 + x_4\}) \\
= & \{x_2 = x_3 + x_4\} \text{ (fix point)}
\end{aligned}$$

5. For the program:

1.  $\text{qs}([], []) \leftarrow$ .
2.  $\text{qs}([X|L], S) \leftarrow \text{part}(X, L, L1, L2),$   
 $\quad \text{qs}(L1, S1), \text{qs}(L2, S2),$   
 $\quad \text{append}(S1, [X|S2], S).$
3.  $\text{part}(X, [], [], []) \leftarrow$ .
4.  $\text{part}(X, [Y|L], [Y|L1], L2) \leftarrow X \geq Y, \text{part}(X, L, L1, L2).$
5.  $\text{part}(X, [Y|L], L1, [Y|L2]) \leftarrow X < Y, \text{part}(X, L, L1, L2).$
6.  $\text{append}([], L, L) \leftarrow$ .
7.  $\text{append}([X|U], V, [X|W]) \leftarrow \text{append}(U, V, W).$

- compute the abstract OLDT tree using the  $\mathcal{G}$  abstraction for ground terms and for the call  $\text{qs}(\mathcal{G}, S)$ .

Using a separate tree for each call:

- The initial call  $\text{qs}(\mathcal{G}, S)$ :

$$\begin{array}{l}
\leftarrow \text{qs}(\mathcal{G}, S) \\
\downarrow \qquad \searrow \\
(1)\{\mathcal{G}\} \qquad (2)\{X/\mathcal{G}, L^1/\mathcal{G}, S^1/S\} \\
\sqcap \qquad \leftarrow \text{part}(\mathcal{G}, \mathcal{G}, L1, L2), \text{qs}(L1, S1), \text{qs}(L2, S2), \text{append}(S1, [\mathcal{G}|S2], S) \\
\text{qs}(\mathcal{G}, \mathcal{G}) \leftarrow (8) \quad (9)\{L1/\mathcal{G}, L2/\mathcal{G}\} \\
\qquad \qquad \qquad \leftarrow \text{qs}(\mathcal{G}, S1), \text{qs}(\mathcal{G}, S2), \text{append}(S1, [\mathcal{G}|S2], S) \\
\qquad \qquad \qquad (8)\{S1/\mathcal{G}\} \\
\qquad \qquad \qquad \leftarrow \text{qs}(\mathcal{G}, S2), \text{append}(\mathcal{G}, [\mathcal{G}|S2], S) \\
\qquad \qquad \qquad (8)\{S2/\mathcal{G}\} \\
\qquad \qquad \qquad \leftarrow \text{append}(\mathcal{G}, \mathcal{G}, S) \\
\qquad \qquad \qquad (10)\{S/\mathcal{G}\} \\
\qquad \qquad \qquad \sqcap \\
\qquad \qquad \qquad \text{qs}(\mathcal{G}, \mathcal{G}) (= 8)
\end{array}$$

– The call  $part(\mathcal{G}, \mathcal{G}, L1, L2)$ :

$$\begin{array}{ccc}
& \leftarrow part(\mathcal{G}, \mathcal{G}, L1, L2) & \\
\swarrow & \downarrow & \searrow \\
(3)\{L1/\mathcal{G}, L2/\mathcal{G}\} & (4)\{X^1/\mathcal{G}, Y^1/\mathcal{G}, L^1/\mathcal{G}, L1/[Y^1|L1^1], L2^1/L2\} & (5) \text{ similar to (4)} \\
\Box & \leftarrow \mathcal{G} > \mathcal{G}, part(\mathcal{G}, \mathcal{G}, [Y^1|L1^1], L2) & \text{omitted} \\
part(\mathcal{G}, \mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow (9) & \text{builtin, no bindings} & \\
& \leftarrow part(\mathcal{G}, \mathcal{G}, [Y^1|L1^1], L2) & \\
& (9)\{Y^1/\mathcal{G}, L1^1/\mathcal{G}, L2/\mathcal{G}\} & \\
& \Box & \\
& part(\mathcal{G}, \mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow (= 9) & 
\end{array}$$

– The call  $append(\mathcal{G}, \mathcal{G}, S)$ :

$$\begin{array}{ccc}
\leftarrow append(\mathcal{G}, \mathcal{G}, S) & & \\
\downarrow & \searrow & \\
(6)\{S/\mathcal{G}\} & (7) \{X^1/\mathcal{G}, U^1/\mathcal{G}, V^1/\mathcal{G}, S/[G|W^1]\} & \\
\Box & \leftarrow append(\mathcal{G}, \mathcal{G}, W^1) & \\
append(\mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow (10) & (10)\{W^1/\mathcal{G}\} & \\
& \Box & \\
& append(\mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow (= 10) & 
\end{array}$$

- Give the set of abstract atoms being called during execution (abstract call patterns for the query).

The call patterns are  $qs(\mathcal{G}, S)$ ,  $part(\mathcal{G}, \mathcal{G}, L1, L2)$ ,  $append(\mathcal{G}, \mathcal{G}, S)$ ,  $\mathcal{G} > \mathcal{G}$  and  $\mathcal{G} \leq \mathcal{G}$ .

6. For the program:

1.  $qs([], []) \leftarrow$ .
2.  $qs([X|L], S) \leftarrow part(X, L, L1, L2),$   
 $qs(L1, S1), qs(L2, S2),$   
 $append(S1, [X|S2], S).$
3.  $part(X, [], [], []) \leftarrow$ .
4.  $part(X, [Y|L], [Y|L1], L2) \leftarrow X \geq Y, part(X, L, L1, L2).$
5.  $part(X, [Y|L], L1, [Y|L2]) \leftarrow X < Y, part(X, L, L1, L2).$
6.  $append([], L, L) \leftarrow$ .
7.  $append([X|U], V, [X|W]) \leftarrow append(U, V, W).$

- compute the abstract LSLDT tree using the  $\mathcal{G}$  abstraction for ground terms and for the call  $qs(\mathcal{G}, S)$ .

– toplevel:

$$\leftarrow qs(L, S); \{L = \mathcal{G}\}$$

Using (8):  $\Box, \{L = \mathcal{G}, S = \mathcal{G}\}$

– Derivation for clause (1) with  $\{L = \mathcal{G}\}$

$$qs(L, S) \leftarrow L = [], S = []; \{L = \mathcal{G}\}$$

$$qs(L, S) \leftarrow S = []; \{L = \mathcal{G}\}$$

$$qs(L, S) \leftarrow; \{L = \mathcal{G}, S = \mathcal{G}\}$$

$$\text{i.e. lemma (8): } qs(\mathcal{G}, \mathcal{G}) \leftarrow$$

– Derivation for clause (2) with  $\{L = \mathcal{G}\}$

$$qs(L, S) \leftarrow L = [X|Ls], part(X, Ls, L1, L2), qs(L1, S1), qs(L2, S2), append(S1, [X|S2], S); \{L = \mathcal{G}\}$$

$qs(L, S) \leftarrow part(X, Ls, L1, L2), qs(L1, S1), qs(L2, S2), append(S1, [X|S2], S); \{L = \mathcal{G}, X = \mathcal{G}, Ls = \mathcal{G}\}$

Using (9):

$qs(L, S) \leftarrow qs(L1, S1), qs(L2, S2), append(S1, [X|S2], S); \{L = \mathcal{G}, X = \mathcal{G}, L1 = \mathcal{G}, L2 = \mathcal{G}\}$

Using (8):

$qs(L, S) \leftarrow qs(L2, S2), append(S1, [X|S2], S); \{L = \mathcal{G}, X = \mathcal{G}, L2 = \mathcal{G}, S1 = \mathcal{G}\}$

Using (8):

$qs(L, S) \leftarrow append(S1, [X|S2], S); \{L = \mathcal{G}, X = \mathcal{G}, S1 = \mathcal{G}, S2 = \mathcal{G}\}$

Using (10):

$qs(L, S) \leftarrow; \{L = \mathcal{G}, S = \mathcal{G}\}$

which yields the same lemma as (8)

If not, one would have to replace (8) by the upperbound of (8) and the new lemma (that would require to introduce an abstraction for “any-term”) and redo the computations where (8) was used.

- Derivation for clause (3) with  $\{X = \mathcal{G}, L = \mathcal{G}\}$ 

$part(X, L, L1, L2) \leftarrow L = [], L1 = [], L2 = []; \{X = \mathcal{G}, L = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow L1 = [], L2 = []; \{X = \mathcal{G}, L = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow L2 = []; \{X = \mathcal{G}, L = \mathcal{G}, L1 = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow; \{X = \mathcal{G}, L = \mathcal{G}, L1 = \mathcal{G}, L2 = \mathcal{G}\}$   
i.e. Lemma (9):  $part(\mathcal{G}, \mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow$
- Derivation for clause (4) with  $\{X = \mathcal{G}, L = \mathcal{G}\}$ 

$part(X, L, L1, L2) \leftarrow L = [Y|Ls], L1 = [Y|L1s], X \geq Y, part(X, L, L1s, L2); \{X = \mathcal{G}, L = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow L1 = [Y|L1s], X \geq Y, part(X, L, L1s, L2); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow X \geq Y, part(X, L, L1s, L2); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L1 = [\mathcal{G}|L1s]\}$   
built-in, no changes in abstract state  
 $part(X, L, L1, L2) \leftarrow part(X, L, L1s, L2); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L1 = [\mathcal{G}|L1s]\}$   
Using (9):  
 $part(X, L, L1, L2) \leftarrow; \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L1 = \mathcal{G}, L1s = \mathcal{G}, L2 = \mathcal{G}\}$   
which yields the same lemma as (9)
- Derivation for clause (5) with  $\{X = \mathcal{G}, L = \mathcal{G}\}$ 

$part(X, L, L1, L2) \leftarrow L = [Y|Ls], L2 = [Y|L2s], X < Y, part(X, L, L1, L2s); \{X = \mathcal{G}, L = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow L2 = [Y|L2s], X < Y, part(X, L, L1, L2s); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}\}$   
 $part(X, L, L1, L2) \leftarrow X < Y, part(X, L, L1, L2s); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L2 = [\mathcal{G}|L2s]\}$   
built-in, no changes in abstract state  
 $part(X, L, L1, L2) \leftarrow part(X, L, L1, L2s); \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L2 = [\mathcal{G}|L2s]\}$   
Using (9):  
 $part(X, L, L1, L2) \leftarrow; \{X = \mathcal{G}, L = \mathcal{G}, Y = \mathcal{G}, Ls = \mathcal{G}, L2 = \mathcal{G}, L2s = \mathcal{G}, L1 = \mathcal{G}\}$   
which yields the same lemma as (9)
- Derivation for clause (6) with  $\{L1 = \mathcal{G}, L2 = \mathcal{G}\}$ 

$append(L1, L2, L3) \leftarrow L1 = [], L2 = L3; \{L1 = \mathcal{G}, L2 = \mathcal{G}\}$

$append(L1, L2, L3) \leftarrow L2 = L3; \{L1 = \mathcal{G}, L2 = \mathcal{G}\}$   
 $append(L1, L2, L3) \leftarrow; \{L1 = \mathcal{G}, L2 = \mathcal{G}, L3 = \mathcal{G}\}$   
 i.e. lemma (10):  $append(\mathcal{G}, \mathcal{G}, \mathcal{G}) \leftarrow$   
 – Derivation for clause (7) with  $\{L1 = \mathcal{G}, L2 = \mathcal{G}\}$   
 $append(L1, L2, L3) \leftarrow L1 = [X|U], L3 = [X|W], append(U, V, W); \{L1 = \mathcal{G}, L2 = \mathcal{G}\}$   
 $append(L1, L2, L3) \leftarrow L3 = [X|W], append(U, V, W); \{L1 = \mathcal{G}, L2 = \mathcal{G}, X = \mathcal{G}, U = \mathcal{G}\}$   
 $append(L1, L2, L3) \leftarrow append(U, V, W); \{L1 = \mathcal{G}, L2 = \mathcal{G}, X = \mathcal{G}, U = \mathcal{G}\}, L3 = [\mathcal{G}|W]\}$   
 Using (10):  
 $append(L1, L2, L3) \leftarrow; \{L1 = \mathcal{G}, L2 = \mathcal{G}, X = \mathcal{G}, U = \mathcal{G}\}, L3 = \mathcal{G}\}$   
 which yields the same lemma as (10)

(Pieces can be plugged together in an AND-OR graph)

- Give the set of abstract atoms being called during execution (abstract call-patterns for the query).

The call patterns are  $qs(\mathcal{G}, S)$ ,  $part(\mathcal{G}, \mathcal{G}, L1, L2)$ ,  $append(\mathcal{G}, \mathcal{G}, S)$ ,  $\mathcal{G} > \mathcal{G}$  and  $\mathcal{G} \leq \mathcal{G}$ .

7. For the program

$qs([], []) \leftarrow.$   
 $qs([X|L], S) \leftarrow part(X, L, L1, L2),$   
 $qs(L1, S1), qs(L2, S2),$   
 $append(S1, [X|S2], S).$   
 $part(X, [], [], []) \leftarrow.$   
 $part(X, [Y|L], [Y|L1], L2) \leftarrow X \geq Y, part(X, L, L1, L2).$   
 $part(X, [Y|L], L1, [Y|L2]) \leftarrow X < Y, part(X, L, L1, L2).$   
 $append([], L, L) \leftarrow.$   
 $append([X|U], V, [X|W]) \leftarrow append(U, V, W).$

use the automated approach to prove left-termination of the calls abstracted by  $qs(\mathcal{G}, S)$ .

- To determine the calls activated by a call abstracted by  $qs(\mathcal{G}, S)$ , see exercise 5 or 6. The calls are those abstracted by  $qs(\mathcal{G}, S)$ ,  $part(\mathcal{G}, \mathcal{G}, L1, L2)$ ,  $append(\mathcal{G}, \mathcal{G}, S)$ ,  $\mathcal{G} > \mathcal{G}$  and  $\mathcal{G} \leq \mathcal{G}$ .
- Termination for the calls abstracted by  $\mathcal{G} > \mathcal{G}$  and  $\mathcal{G} \leq \mathcal{G}$  is trivial (built-ins).
- **part/4** is recurrent with respect to the set of call abstracted by  $part(\mathcal{G}, \mathcal{G}, L1, L2)$  (see exercise 4 in the part on termination analysis), hence these calls are terminating.
- **append/3** is recurrent with respect to the set of call abstracted by  $append(\mathcal{G}, \mathcal{G}, S)$  (see exercise 4 in the part on termination analysis), hence these calls are terminating.
- An argument-size relationship for **part/4** is proven in exercise 4; using this relationship, one can prove that **part/4** is acceptable with respect to the set of calls abstracted by  $part(\mathcal{G}, \mathcal{G}, L1, L2)$  (as shown in exercise 4 in the part on termination analysis), hence these calls are left-terminating.
- Hence calls abstracted by  $qs(\mathcal{G}, S)$  are left-terminating.