On the Verification of a Real Prolog Program: `crossword`

Roberto Bruni Marco Pistore

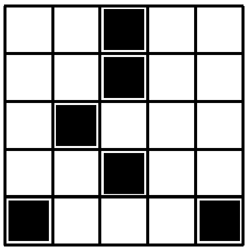
March 1996

1 The problem: the 4^{th} exercise of ILPS'94

An empty crossword puzzle is exemplified by the facts:

```
dimension(5).  % it is always a square crossword puzzle
black(1,3).  % meaning: square 1,3 is black
black(2,3).  black(3,2).  black(4,3).  black(5,1).  black(5,5).
```

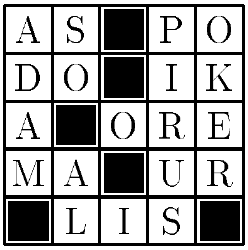
These example facts represent the empty puzzle:



A list of words is also given, for example:

```
words([do,ore,ma,lis,ur,as,po,so,pirus,oker,al,adam,ik]).
```

All words in the list have at least two characters. The goal is to fill the puzzle with all the words exactly once, or to fail if that is impossible. For instance, filling the exemplified puzzle with the above words gives:



The program should give the solution as the output parameter of `crossword/1` and in the form of a list of words in the following order:

```
[as,po,do,ik,ore,ma,ur,lis,adam,so,al,pirus,oker]
```

i.e., first all the horizontal words, row by row and in a row as they occur from left to right, then all the vertical words, column by column.

If there is more than one solution, the program should produce them all by backtracking.

2 A solution

We present a solution to the problem, together with a specification (pre- and post-conditions and level mappings), a moding and a typing for each predicate.

We use the following types:

- $GaeN$ are ground arithmetic expressions corresponding to natural numbers; if $N \in GaeN$ then we use N itself to refer to $val(N)$.
- $Char$ are characters.
- $Atom$ are prolog atoms. For sake of simplicity, in what follows we identify an atom with the list of its characters.
- $Term$ are generic prolog terms.
- If $Type$ is a type, $TypeList$ are the finite lists of elements of type $Type$.
- $Dir = \{\texttt{orz}, \texttt{vrt}\}$ are the directions in the crossword.
- $FilaDim = \{\texttt{fila}(D, Y, X, L) \mid D \in Dir \wedge Y, X, L \in GaeN\}$ represents filling boxes of the crossword with direction D, starting cell (Y, X) and with length L.
- $Fila = \{\texttt{fila}(D, Y, X, Ws) \mid D \in Dir \wedge Y, X \in GaeN \wedge Ws \in AtomList\}$ represents filling boxes of the crossword with direction D, starting cell (Y, X); Ws represents the set of the atoms which can be assigned to the particular box.
- $Res = \{\texttt{res}(D, Y, X, W) \mid D \in Dir \wedge Y, X \in GaeN \wedge W \in Atom\}$ represents boxes of the crossword with direction D, starting cell (Y, X) and filled with atom W.
- $Cell = \{\texttt{cell}(Y, X, Ch) \mid Y, X \in GaeN \wedge Ch \in Char\}$.
- $Cross = CellList$.

In the program we omit pre- and post-conditions when they are obvious. Moreover, we use some shorthands:

- If L is a list, $\{\!|L|\!\}$ is the (multi)set of the elements of L.
- If SC is a set of $Cell$, SF is a set of $Fila$ and SR is a set of Res, then $SR \in sol(SC, SF)$ if and only if either $SF = \emptyset = SR$, or whenever $R_0 = \texttt{res}(D, Y, X, W) \in SR$ then:
  1. $F_0 = \texttt{fila}(D, Y, X, Ws) \in SF$ and $W \in \{\!|Ws|\!\}$,
  2. $SR - \{R_0\} \in sol(SC', SF')$, where $SC' = assign(SC, W, D, Y, X)$ and $SF' = \{\texttt{fila}(D, Y, X, Ws') \mid \texttt{fila}(D, Y, X, Ws) \in SF - \{F_0\} \wedge \{\!|Ws'|\!\} = filter(D, Y, X, \{\!|Ws|\!\} - \{W\}, SC')\}$.
- If SC is a set of $Cell$, $W = [Ch_1, \ldots, Ch_n] \in Atom$, $D \in Dir$ and $Y, X \in GaeN$ then

$$assign(SC, W, D, Y, X) = \begin{cases} SC \cup \{\texttt{cell}(Y, X+i, Ch_i) \mid i = 0..n\} & \text{if } D = \texttt{orz} \\ SC \cup \{\texttt{cell}(Y+i, X, Ch_i) \mid i = 0..n\} & \text{if } D = \texttt{vrt} \end{cases}$$

- If $Y, X \in GaeN$, SW is a set of $Atom$ and SC is a set of $Cell$, then

  $$filter(\texttt{orz}, Y, X, SW, SC) = \{[Ch_0, \ldots, Ch_n] \in SW \mid \texttt{cell}(Y, X+i, Ch) \in SC, i = 0..n \Longrightarrow Ch = Ch_i\}$$

  and analogously for $\texttt{vrt}$.
- If SC is a set of $Cell$ then $consistent(SC)$ holds iff $\texttt{cell}(Y, X, Ch), \texttt{cell}(Y, X, Ch') \in SC$ imply $Ch = Ch'$.

- If SF is a set of $Fila$ and SC is a set of $Cell$ then $compatible(SF, SC)$ is true if and only if $\texttt{fila}(\texttt{orz}, Y, X, Ws) \in SF$, $[Ch_0, \ldots Ch_n] \in \{\!|Ws|\!\}$, $\texttt{cell}(Y, X+i, Ch) \in SC$, $i = 0..n$ imply $Ch = Ch_i$; analogously for `vrt`.
- If $R = \texttt{res}(D, Y, X, W)$ and $R' = \texttt{res}(D', Y', X', W')$ then $R <_r R'$ iff one of the following conditions holds:
  - $D = \texttt{orz}$ and $D' = \texttt{vrt}$; or
  - $D = D' = \texttt{orz}$ and $Y < Y'$; or
  - $D = D' = \texttt{orz}$, $Y = Y'$ and $X < X'$; or
  - $D = D' = \texttt{vrt}$ and $X < X'$; or
  - $D = D' = \texttt{vrt}$, $X = X'$ and $Y < Y'$.
- If C is the empty crossword correspoding to dimension n and set of black cells $black$, and $Y, X = 1..n+1$ then $orz(C, Y, X)$ gives the ordered list (left-to-right and top-down) of $FilaDim$ which describes the contiguous horizontal empty areas between black cells (and margins), from position (Y, X) towards the end of the crossword. For instance, if C is the empty puzzle previously described, then

  $$\begin{aligned} orz(C,3,2) = [&\texttt{filadim}(\texttt{orz},3,2,0), \texttt{filadim}(\texttt{orz},3,3,3), \texttt{filadim}(\texttt{orz},4,1,2),\\ &\texttt{filadim}(\texttt{orz},4,4,2), \texttt{filadim}(\texttt{orz},5,1,0), \texttt{filadim}(\texttt{orz},5,2,3),\\ &\texttt{filadim}(\texttt{orz},5,6,0), \texttt{filadim}(\texttt{orz},6,1,0)] \end{aligned}$$

  As this example shows, in the list do appear some areas on length 0 when two consecutive black cells (or a black cell and a margin) are present in the crossword and an area of length 0 for the $(n+1)^{\text{th}}$ row.
- $vrt(C, Y, X)$ is defined similarly.
- If FDS is a set of $FilaDim$ then $useful(FDS) = \{\texttt{filadim}(D, Y, X, L) \mid L > 1\}$.
- A *solution* of the puzzle corresponding to crossword C and words ws is an ordered list Ws of words that fill the crossword and such that $\{\!|Ws|\!\} \subseteq ws$; it is an *exact solution* if $\{\!|Ws|\!\} = ws$.

Suppose that crossword C_0 has dimension n_0 and $black_0$ as set of black cells; ws_0 is the set of words which should be used to fill C_0. We use P_0 to denote the puzzle corresponding to crossword C_0 and set of words ws_0.

```
%
% Crossword solver
%
% by Roberto BRUNI and Marco PISTORE
%
% (C) 1995
%
```

% dimension$(- : GaeN)$
% $Pre = \{\texttt{dimension}(N)\}$
% $Post = \{\texttt{dimension}(n_0)\}$

```
dimension(5).
```

% black$(+ : GaeN, + : GaeN)$
% $Post = \{\texttt{black}(Y, X) \mid (Y, X) \in black_0\}$

```
black(1,3).
black(2,3).
black(3,2).
```

```
black(4,3).
black(5,1).
black(5,5).
```

% $\texttt{words}(- : AtomList)$
% $Pre = \{\texttt{words}(Ws)\}$
% $Post = \{\texttt{words}(Ws) \mid \{\!\!\{Ws\}\!\!\} = ws_0\}$

```
words([do,ore,ma,lis,ur,as,po,so,pirus,oker,al,adam,ik]).
```

% $\texttt{crossword}(- : AtomList)$
% $Post = \{\texttt{crossword}(Ws) \mid Ws$ is an exact solution of P_0 $\}$

```
crossword(Ws) :-
    build_file(Fs),
    words(Ws1), ln(Ws1, L), ln(Fs, L),
    build_cross(Fs, [], Rs),
    ordina_res(Rs, Ws).
```

% $\texttt{build\_file}(- : FilaList)$
% $Post = \{\texttt{build\_file}(Fs) \mid \{R_1, \ldots, R_n\} \in sol(\emptyset, \{\!\!\{Fs\}\!\!\}), R_i <_r R_{i+1},$
% $\qquad R_i = \texttt{res}(D_i, Y_i, X_i, W_i) \Longrightarrow [W_1, \ldots, W_n]$ is a solution of $P_0\}$

```
build_file(Fs) :-
    build_orz(1,1,FDs1),
    build_vrt(1,1,FDs2),
    app(FDs1,FDs2,FDs),
    assoc_words(FDs, Fs).
```

% $\texttt{build\_orz}(+ : GaeN, + : GaeN, - : FilaDimList)$
% $Pre = \{\texttt{build\_orz}(Y, X, FDs) \mid Y, X \in [1..n_0 + 1]\}$
% $Post = \{\texttt{build\_orz}(Y, X, FDs) \mid Y, X \in [1..n_0 + 1] \wedge FDs = orz(C_0, Y, X)\}$
% $|\texttt{build\_orz}(Y, X, FDs)| = (n_0 + 1 - X) + (n_0 + 1 - Y)(n_0 + 1)$

```
build_orz(M,X,[filadim(orz,M,X,0)]) :-
    dimension(N), M =:= N+1.

build_orz(Y,M,[filadim(orz,Y,M,0) | FDs]) :-
    dimension(N), M =:= N+1, Y =< N, Y1 is Y + 1,
    build_orz(Y1,1,FDs).

build_orz(Y,X,[filadim(orz,Y,X,L1) | FDs]) :-
    dimension(N), X=<N, Y =<N,
    \+black(Y,X), X1 is X+1,
    build_orz(Y,X1,[filadim(_,_,_,L) | FDs]),
    L1 is L+1.

build_orz(Y,X,[filadim(orz,Y,X,0) | FDs]) :-
    dimension(N), X=<N, Y =<N,
    black(Y,X), X1 is X+1,
    build_orz(Y,X1,FDs).
```

% $\texttt{build\_vrt}(+ : GaeN, + : GaeN, - : FilaDimList)$
% $Pre = \{\texttt{build\_vrt}(Y, X, FDs) \mid Y, X \in [1..n_0 + 1]\}$
% $Post = \{\texttt{build\_vrt}(Y, X, FDs) \mid Y, X \in [1..n_0 + 1] \wedge FDs = vrt(C_0, Y, X)\}$
% $|\texttt{build\_vrt}(Y, X, FDs)| = (n_0 + 1 - Y) + (n_0 + 1 - X)(n_0 + 1)$

```
build_vrt(Y,M,[filadim(vrt,Y,M,0)]) :-
    dimension(N), M =:= N+1.

build_vrt(M,X,[filadim(vrt,M,X,0) | FDs]) :-
    dimension(N), M =:= N+1, X =< N, X1 is X + 1,
```

```
    build_vrt(1,X1,FDs).
build_vrt(Y,X,[filadim(vrt,Y,X,L1) | FDs]) :-
    dimension(N), X=<N, Y =<N,
    \+black(Y,X), Y1 is Y+1,
    build_vrt(Y1,X,[filadim(_,_,_,L) | FDs]),
    L1 is L+1.
build_vrt(Y,X,[filadim(vrt,Y,X,0) | FDs]) :-
    dimension(N), X=<N, Y =<N,
    black(Y,X), Y1 is Y+1,
    build_vrt(Y1,X,FDs).
```

% $\texttt{assoc\_words}(+ : \mathit{FilaDimList}, - : \mathit{FilaList})$
% $\mathit{Post} = \{\texttt{assoc\_words}(FDs, Fs) \mid \mathit{useful}(\{\!|FDs|\!\}) = \{\texttt{filadim}(D_i, Y_i, X_i, L_i) \mid i = 1..n\} \wedge$
% $\qquad \{\!|Fs|\!\} = \{\texttt{fila}(D_i, Y_i, X_i, Ws_i) \mid i = 1..n\} \wedge \{\!|Ws_i|\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$
% $|\texttt{assoc\_words}(FDs, Fs)| = \|FDs\|$

```
assoc_words([],[]).
assoc_words([filadim(_,_,_,0) | FDs], Fs) :-
    assoc_words(FDs,Fs).
assoc_words([filadim(_,_,_,1) | FDs], Fs) :-
    assoc_words(FDs,Fs).
assoc_words([filadim(D,Y,X,L) | FDs], [fila(D,Y,X,Ws1) | Fs]) :-
    L > 1,
    assoc_words(FDs,Fs),
    words(Ws), words_of_ln(Ws,L,Ws1).
```

% $\texttt{words\_of\_ln}(+ : \mathit{AtomList}, + : \mathit{GaeN}, - : \mathit{AtomList})$
% $\mathit{Post} = \{\texttt{words\_of\_ln}(Ws, L, Ws1) \mid \{\!|Ws1|\!\} = \{w \in \{\!|Ws|\!\} \mid \|w\| = L\}\}$
% $|\texttt{words\_of\_ln}(Ws, L, Ws1)| = \|Ws\|$

```
words_of_ln([],_,[]).
words_of_ln([W | WS],L,WS1) :-
    atom_chars(W,W1),
    ln(W1,M), M =\= L,
    words_of_ln(WS,L,WS1).
words_of_ln([W | WS],L,[W | WS1]) :-
    atom_chars(W,W1),
    ln(W1,L),
    words_of_ln(WS,L,WS1).
```

% $\texttt{ordina\_res}(+ : \mathit{ResList}, - : \mathit{AtomList})$
% $\mathit{Post} = \{\texttt{ordina\_res}(Rs, Ws) \mid \{\!|Rs|\!\} = \{R_1, \ldots, R_n\} \wedge R_i <_r R_{i+1} \wedge$
% $\qquad R_i = \texttt{res}(D_i, Y_i, X_i, W_i) \wedge Ws = [W_1, \ldots, W_n]\}$

```
ordina_res(Rs,Ws) :-
    qs_file(Rs,Rs1),
    only_words(Rs1,Ws).
```

% $\texttt{qs\_file}(+ : \mathit{ResList}, - : \mathit{ResList})$
% $\mathit{Post} = \{\texttt{qs\_file}(Rs, Rs1) \mid \{\!|Rs|\!\} = \{R_1, \ldots, R_n\} \wedge R_i <_r R_{i+1} \wedge Rs1 = [R_1, \ldots, R_n]\}$
% $|\texttt{qs\_file}(Rs, Rs1)| = \|Rs\|$

```
qs_file([],[]).
qs_file([R | Rs], Rs1) :-
    part_file(R, Rs, Littles, Bigs),
    qs_file(Littles,LRs),
```

```
    qs_file(Bigs,BRs),
    app(LRs, [R | BRs], Rs1).
```

% part_file$(+ : Res, + : ResList, - : ResList, - : ResList)$
% $Post = \{$part_file$(R, Rs, LRs, BRs) \mid \{\!\!\{Rs\}\!\!\} = \{\!\!\{LRs\}\!\!\} \cup \{\!\!\{BRs\}\!\!\} \wedge$
% $\qquad R1 \in LRs \Longrightarrow R1 <_r R \wedge R1 \in BRs \Longrightarrow R <_r R1\}$
% $|$part_file$(R, Rs, LRs, BRs)| = \|Rs\|$

```
part_file(_,[],[],[]).
part_file(R,[R1 | Rs],[R1 | LRs],BRs) :-
    greater_fila(R,R1),
    part_file(R,Rs,LRs,BRs).
part_file(R,[R1 | Rs],LRs,[R1 | BRs]) :-
    \+ greater_fila(R,R1),
    part_file(R,Rs,LRs,BRs).
```

% greater_fila$(+ : Res, + : Res)$
% $Post = \{$greater_fila$(R, R1) \mid R1 <_r R\}$

```
greater_fila(res(vrt,_,_,_),res(orz,_,_,_)).
greater_fila(res(orz,Y,_,_),res(orz,Y1,_,_)) :- Y > Y1.
greater_fila(res(orz,Y,X,_),res(orz,Y,X1,_)) :- X > X1.
greater_fila(res(vrt,_,X,_),res(vrt,_,X1,_)) :- X > X1.
greater_fila(res(vrt,Y,X,_),res(vrt,Y1,X,_)) :- Y > Y1.
```

% only_words$(+ : ResList, - : AtomList)$
% $Post = \{$only_words$(Rs, Ws) \mid Rs = [R_1, \ldots, R_n] \wedge R_i = \mathtt{res}(D_i, Y_i, X_i, W_i) \wedge$
% $\qquad Ws = [W_1, \ldots, W_n]\}$
% $|$only_words$(Rs, Ws)| = \|Rs\|$

```
only_words([],[]).
only_words([res(_,_,_,W)|Rs],[W|Ws]) :-
    only_words(Rs,Ws).
```

% build_cross$(+ : FilaList, + : Cross, - : ResList)$
% $Pre = \{$build_cross$(Fs, C, Rs) \mid consistent(\{\!\!\{C\}\!\!\}) \wedge compatible(\{\!\!\{Fs\}\!\!\}, \{\!\!\{C\}\!\!\})\}$
% $Post = \{$build_cross$(Fs, C, Rs) \mid consistent(\{\!\!\{C\}\!\!\}) \wedge compatible(\{\!\!\{Fs\}\!\!\}, \{\!\!\{C\}\!\!\}) \wedge$
% $\qquad \{\!\!\{Rs\}\!\!\} \in sol(\{\!\!\{C\}\!\!\}, \{\!\!\{Fs\}\!\!\})\}$
% $|$build_cross$(Fs, C, Rs)| = \|Fs\|$

```
build_cross([],_,[]).
build_cross([F | Fs], C, [res(D,Y,X,W) | Rs]) :-
    min_fila([F | Fs], fila(D,Y,X,Ws), Fs1),
    member(W, Ws),
    assegna(D,Y,X,W,C,C1),
    aggiorna(Fs1,W,C1,Fs2),
    build_cross(Fs2,C1,Rs).
```

% assegna$(+ : Dir, + : GaeN, + : GaeN, + : Atom, + : Cross, - : Cross)$
% $Post = \{$assegna$(D, Y, X, W, C, C1) \mid \{\!\!\{C1\}\!\!\} = assign(\{\!\!\{C\}\!\!\}, W, D, Y, X)\}$

```
assegna(D,Y,X,W,C,C1) :-
    atom_chars(W,W1),
    assegna2(D,Y,X,W1,C,C1).
```

% assegna2$(+ : Dir, + : GaeN, + : GaeN, + : CharList, + : Cross, - : Cross)$
% $Post = \{$assegna2$(D, Y, X, W, C, C1) \mid \{\!\!\{C1\}\!\!\} = assign(\{\!\!\{C\}\!\!\}, W, D, Y, X)\}$

% $|\texttt{assegna2}(D, Y, X, W, C, C1)| = ||W||$

```
assegna2(_,_,_,[],C,C).
assegna2(orz,Y,X,[Ch|W],C,[cell(Y,X,Ch) | C1]) :-
    X1 is X+1,
    assegna2(orz,Y,X1,W,C,C1).
assegna2(vrt,Y,X,[Ch|W],C,[cell(Y,X,Ch) | C1]) :-
    Y1 is Y+1,
    assegna2(vrt,Y1,X,W,C,C1).
```

% $\texttt{aggiorna}(+ : FilaList, + : Atom, + : Cross, - : FilaList)$
% $Pre = \{\texttt{aggiorna}(Fs, W, C, Fs1) \mid consistent(\{\![C]\!\})\}$
% $Post = \{\texttt{aggiorna}(Fs, W, C, Fs1) \mid consistent(\{\![C]\!\}) \wedge \{\![Fs]\!\} = \{F_1, \ldots, F_n\} \wedge \{\![Fs1]\!\} = \{F'_1, \ldots F'_n\} \wedge$
% $\quad F_i = \texttt{fila}(D_i, Y_i, X_i, Ws_i) \wedge F'_i = \texttt{fila}(D_i, Y_i, X_i, Ws'_i) \wedge$
% $\quad \{\![Ws'_i]\!\} = filter(D_i, Y_i, X_i, \{\![Ws_i]\!\} - \{W\}, \{\![C]\!\})\}$
% $|\texttt{aggiorna}(Fs, W, C, Fs1)| = ||Fs||$

```
aggiorna([],_,_,[]).
aggiorna([fila(D,Y,X,Ws)|Fs1],W,C,[fila(D,Y,X,Ws2)|Fs2]) :-
    minus_el(Ws,W,Ws1),
    filtra_cross(D,Y,X,Ws1,C,Ws2),
    aggiorna(Fs1,W,C,Fs2).
```

% $\texttt{minus\_el}(+ : AtomList, + : Atom, - : AtomList)$
% $Post = \{\texttt{minus\_el}(Ws, W, Ws1) \mid \{\![Ws1]\!\} = \{\![Ws]\!\} - \{W\}\}$
% $|\texttt{minus\_el}(Ws, W, Ws1)| = ||Ws||$

```
minus_el([],_,[]).
minus_el([W|Ws],W,Ws).
minus_el([W|Ws],W1,[W|Ws1]) :-
    W \== W1,
    minus_el(Ws,W1,Ws1).
```

% $\texttt{filtra\_cross}(+ : Dir, + : GaeN, + : GaeN, + : AtomList, + : Cross, - : AtomList)$
% $Pre = \{\texttt{filtra\_cross}(D, Y, X, Ws, C, Ws1) \mid consistent(\{\![C]\!\})\}$
% $Post = \{\texttt{filtra\_cross}(D, Y, X, Ws, C, Ws1) \mid consistent(\{\![C]\!\}) \wedge \{\![Ws1]\!\} = filter(D, Y, X, \{\![Ws]\!\}, \{\![C]\!\})\}$
% $|\texttt{filtra\_cross}(D, Y, S, Ws, C, Ws1)| = ||Ws||$

```
filtra_cross(_,_,_,[],_,[]).
filtra_cross(D,Y,X,[W|Ws],C,[W|Ws1]) :-
    atom_chars(W,W1),
    filtra_cross_aux(D,Y,X,W1,C),
    filtra_cross(D,Y,X,Ws,C,Ws1).
filtra_cross(D,Y,X,[W|Ws],C,Ws1) :-
    atom_chars(W,W1),
    \+ filtra_cross_aux(D,Y,X,W1,C),
    filtra_cross(D,Y,X,Ws,C,Ws1).
```

% $\texttt{filtra\_cross\_aux}(+ : Dir, + : GaeN, + : GaeN, + : CharList, + : Cross)$
% $Pre = \{\texttt{filtra\_cross\_aux}(D, Y, X, W, X) \mid consistent(\{\![C]\!\})\}$
% $Post = \{\texttt{filtra\_cross\_aux}(D, Y, X, W, C) \mid consistent(\{\![C]\!\}) \wedge filter(D, Y, X, \{W\}, \{\![C]\!\}) = \{W\}\}$
% $|\texttt{filtra\_cross\_aux}(D, Y, S, W, C)| = ||W||$

```
filtra_cross_aux(_,_,_,[],_).
filtra_cross_aux(orz,Y,X,[Ch|W],C) :-
    possibly_in(Y,X,Ch,C),
    X1 is X+1,
```

```
    filtra_cross_aux(orz,Y,X1,W,C).
filtra_cross_aux(vrt,Y,X,[Ch|W],C) :-
    possibly_in(Y,X,Ch,C),
    Y1 is Y+1,
    filtra_cross_aux(vrt,Y1,X,W,C).
```

% possibly_in$(+:GaeN, +:GaeN, +:Char, +:Cross)$
% $Pre = \{\texttt{possibly\_in}(Y, X, Ch, C) \mid consistent(\{\!|C|\!\})\}$
% $Post = \{\texttt{possibly\_in}(Y, X, Ch, C) \mid consistent(\{\!|C|\!\}) \wedge \texttt{cell}(Y, X, Ch1) \in \{\!|C|\!\} \Longrightarrow Ch1 = Ch\}$
% $|\texttt{possibly\_in}(Y, X, Ch, C)| = \|C\|$

```
possibly_in(Y,X,Ch,[cell(Y,X,Ch)|_]).
possibly_in(_,_,_,[]).
possibly_in(Y,X,Ch,[cell(Y1,_,_)|C]) :-
    Y =\= Y1,
    possibly_in(Y,X,Ch,C).
possibly_in(Y,X,Ch,[cell(Y,X1,_)|C]) :-
    X =\= X1,
    possibly_in(Y,X,Ch,C).
```

% min_fila$(+:FilaList, -:Fila, -:FilaList)$
% $Post = \{\texttt{min\_fila}(Fs, F, Fs1) \mid \{\!|Fs|\!\} = \{\!|Fs1|\!\} \cup \{F\}\}$
% $|\texttt{min\_fila}(Fs, F, Fs1)| = \|Fs\|$

```
min_fila([F],F,[]).
min_fila([F|Fs],M,[F|Fs1]) :-
    min_fila(Fs,M,Fs1),
    shorter_fila(M,F).
min_fila([F|Fs],F,[M|Fs1]) :-
    min_fila(Fs,M,Fs1),
    \+ shorter_fila(M,F).
```

% shorter_fila$(+:Fila, +:Fila)$
% $Post = \{\texttt{shorter\_fila}(\texttt{fila}(D, Y, X, Ws), \texttt{fila}(D', Y', X', Ws1)) \mid \|Ws\| \leq \|Ws1\|\}$

```
shorter_fila(fila(_,_,_,Ws),fila(_,_,_,Ws1)) :-
    ln(Ws,L), ln(Ws1,L1), L =< L1.
```

% ln$(+:List, -:GaeN)$
% $Post = \{\texttt{ln}(Ls, L) \mid L = \|Ls\|\}$

% member$(-:Atom, +:AtomList)$
% $Post = \{\texttt{member}(A, As) \mid A \in \{\!|As|\!\}\}$

% app$(-:TermList, -:TermList, +:TermList)$
% $Post = \{\texttt{app}(Ls1, Ls2, Ls) \mid Ls = Ls1 \star Ls2\}$

% atom_char$(+:Atom, -:CharList)$
% $Post = \{\texttt{atom\_char}(W, W1) \mid W = W1\}$

% is$(-:GaeN, +:GaeN)$
% $Post = \{\texttt{is}(N, M) \mid N = M\}$

% =<$(+:GaeN, +:GaeN)$
% $Post = \{\texttt{=<}(N, M) \mid N \leq M\}$

% <$(+:GaeN, +:GaeN)$

% $Post = \{\texttt{<}(N, M) \mid N < M\}$

% `=:=`$(+ : GaeN, + : GaeN)$
% $Post = \{\texttt{=:=}(N, M) \mid N = M\}$

% `=\=`$(+ : GaeN, + : GaeN)$
% $Post = \{\texttt{=\textbackslash=}(N, M) \mid N \neq M\}$

% `\==`$(+ : Term, + : Term)$
% $Post = \{\texttt{\textbackslash==}(T, T)\}$

3 Weak correctness

It is easy to see that:

Proposition 1 *The program is well-moded, well-typed, output-linear and non-floundering.*

This is sufficient to prove that:

Corollary 2

- *The occur check can be safely omitted.*
- *The program does not end in error.*

Partial correctness and termination of the program can be derived from the following proof outlines (only the interesting predicates are presented).

```
      {true}
crossword(Ws) :-
          {true}
    build_file(Fs),
          {{R1,...,Rn} ∈ sol(∅, {|Fs|}), Ri <r Ri+1, Ri = res(Di,Yi,Xi,Wi) ⟹
            [W1,...,Wn] is a solution of P0}
          {true}
    words(Ws1),
          {{|Ws1|} = ws0}
          {true}
    ln(Ws1,L),
          {L = ||Ws1||}
          {true}
    ln(Fs,L),
          {L = ||Fs||}
          {consistent(∅) ∧ compatible({|Fs|}, ∅)}
    build_cross(Fs,[],Rs),
          {consistent(∅) ∧ compatible({|Fs|}, ∅) ∧ {|Rs|} ∈ sol(∅, {|Fs|})}
          {true}
    ordina_res(Rs,Ws).
          {{|Rs|} = {R1,...,Rn} ∧ Ri <r Ri+1 ∧ Ri = res(Di,Yi,Xi,Wi) ∧ Ws = [W1,...,Wn]}
      {Ws is an exact solution of P0}
```

Notice that if $[W_1, \ldots, W_n]$ is a solution of P_0 and $n = |ws_0|$ then $[W_1, \ldots, W_n]$ is an exact solution of P_0.

```
      {true}
build_file(Fs) :-
```

$\{1 \in [1..n_0+1]\}$

```
build_orz(1,1,FDs1),
```

$\{FDs1 = orz(C_0,1,1)\}$

$\{1 \in [1..n_0+1]\}$

```
build_vrt(1,1,FDs2),
```

$\{FDs2 = vrt(C_0,1,1)\}$

$\{true\}$

```
app(FDs1,FDs2,FDs),
```

$\{FDs = FDs1 \star FDs2\}$

$\{true\}$

```
assoc_words(FDs,Fs).
```

$\{\{\!\!\{FDs\}\!\!\} = \{\mathtt{filadim}(D_i,Y_i,X_i,L_i) \mid i = 1..n\} \wedge$
$\{\!\!\{Fs\}\!\!\} = \mathtt{fila}(D_i,Y_i,X_i,Ws_i) \mid i = 1..n\} \wedge \{\!\!\{Ws_i\}\!\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{R_1,\ldots,R_n\} \in sol(\emptyset, \{\!\!\{Fs\}\!\!\}),\ R_i <_r R_{i+1},\ R_i = \mathtt{res}(D_i,Y_i,X_i,W_i) \Longrightarrow$
$[W_1,\ldots,W_n]$ is a solution of $P_0\}$

The previous proof outline is based on the following lemma, whose proof is difficult to formalize, although it is conceptually simple.

Lemma 3 *Suppose that:*

1. $useful(\{\!\!\{orz(C_0,1,1)\}\!\!\} \cup \{\!\!\{vrt(C_0,1,1)\}\!\!\}) = \{\mathtt{filadim}(D_i,Y_i,X_i,L_i) \mid i = 1..n\}$,
2. $\{\!\!\{Ws_i\}\!\!\} = \{w \in ws_0 \mid \|w\| = L_i\}$, *and*
3. $\{R_1,\ldots,R_n\} \in sol(\emptyset, \{\!\!\{Fs\}\!\!\})$ *with* $R_i <_r R_{i+1}$ *and* $R_i = \mathtt{res}(D_i,Y_i,X_i,W_i)$.

Then $[W_1,\ldots,W_n]$ *is a solution of* P_0.

$\{M,X \in [1..n_0+1]\}$

```
build_orz(M,X,[filadim(orz,M,X,0)]) :-
```
$\{(n_0+1-X)+(n_0+1-M)(n_0+1)\}$

$\{true\}$

```
dimension(N),
```

$\{N = n_0\}$

$\{true\}$

```
M =:= N+1.
```

$\{M = n_0+1\}$

$\{n_0+1, X \in [1..n_0+1] \wedge [\mathtt{filadim}(\mathtt{orz}, n_0+1, X, 0)] = orz(C_0, n_0+1, X)\}$

$\{Y,M \in [1..n_0+1]\}$

```
build_orz(Y,M,[filadim(orz,Y,M,0) | FDs]) :-
```
$\{(n_0+1-M)+(n_0+1-Y)(n_0+1)\}$

$\{true\}$

```
dimension(N),
```

$\{N = n_0\}$

$\{true\}$

```
M =:= N+1,
```

$\{M = n_0+1\}$

$\{true\}$

```
Y =< N,
```

$\{Y \le n_0\}$

$\{true\}$

```
Y1 is Y+1,
```

$\{Y1 = Y+1\}$

$\{Y+1, 1 \in [1..n_0+1]\}$

```
build_orz(Y1,1,FDs).
```
$\{n_0+(n_0-Y)(n_0+1)\}$

$\{Y+1, 1 \in [1..n_0+1] \wedge FDs = orz(C_0, Y+1, 1)\}$

$\{Y, n_0+1 \in [1..n_0+1] \wedge [\texttt{filadim}(\texttt{orz}, Y, n_0+1, 0)|FDs] = orz(C_0, Y, n_0+1)\}$

$\{Y, X \in [1..n_0+1]\}$

`build_orz(Y,X,[filadim(orz,Y,X,L1) | FDs]) :-` $\{(n_0+1-X)+(n_0+1-Y)(n_0+1)\}$

$\{true\}$

`dimension(N),`

$\{N = n_0\}$

$\{true\}$

`X =< N,`

$\{X \leq n_0\}$

$\{true\}$

`Y =< N,`

$\{Y \leq n_0\}$

$\{true\}$

`\+black(Y,X),`

$\{(Y,X) \notin black_0\}$

$\{true\}$

`X1 is X+1,`

$\{X1 = X+1\}$

$\{Y, X+1 \in [1..n_0+1]\}$

`build_orz(Y,X1,[filadim(D,W,Z,L) | FDs]),` $\{(n_0-X)+(n_0+1-Y)(n_0+1)\}$

$\{Y, X+1 \in [1..n_0+1] \wedge [\texttt{filadim}(\texttt{orz}, W, Z, L)|FDs] = orz(C_0, Y, X+1)\}$

$\{true\}$

`L1 is L+1.`

$\{L1 = L+1\}$

$\{Y, X \in [1..n_0+1] \wedge [\texttt{filadim}(\texttt{orz}, Y, X, L+1)|FDs] = orz(C_0, Y, X)\}$

$\{Y, X \in [1..n_0+1]\}$

`build_orz(Y,X,[filadim(orz,Y,X,0) | FDs]) :-` $\{(n_0+1-X)+(n_0+1-Y)(n_0+1)\}$

$\{true\}$

`dimension(N),`

$\{N = n_0\}$

$\{true\}$

`X =< N,`

$\{X \leq n_0\}$

$\{true\}$

`Y =< N,`

$\{Y \leq n_0\}$

$\{true\}$

`black(Y,X),`

$\{(Y,X) \in black_0\}$

$\{true\}$

`X1 is X+1,`

$\{X1 = X+1\}$

$\{Y, X+1 \in [1..n_0+1]\}$

`build_orz(Y,X1,FDs).` $\{(n_0-X)+(n_0+1-Y)(n_0+1)\}$

$\{Y, X+1 \in [1..n_0+1] \wedge FDs = orz(C_0, Y, X+1)\}$

$\{Y, X \in [1..n_0+1] \wedge [\texttt{filadim}(\texttt{orz}, Y, X, 0)|FDs] = orz(C_0, Y, X)\}$

$\{true\}$

`assoc_words([],[]).` $\{0\}$

$\{[\![\texttt{[]}]\!] = \emptyset\}$

$\{true\}$

`assoc_words([filadim(D,Y,X,0) | FDs],Fs) :-` $\{\|[FDs]\|+1\}$

$\{true\}$

`assoc_words(FDs,Fs).` $\{\|FDs\|\}$

$\{useful(\{\![FDs]\!\}) = \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![Fs]\!\} = \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{useful(\{\![[\texttt{filadim}(D,Y,X,0)|FDs]]\!\}) = \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![Fs]\!\} = \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{true\}$

`assoc_words([filadim(D,Y,X,1) | FDs],Fs) :-` $\{\|FDs\|+1\}$

$\{true\}$

`assoc_words(FDs,Fs).` $\{\|FDs\|\}$

$\{useful(\{\![FDs]\!\}) = \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![Fs]\!\} = \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{useful(\{\![[\texttt{filadim}(D,Y,X,1)|FDs]]\!\}) = \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![Fs]\!\} = \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{true\}$

`assoc_words([filadim(D,Y,X,L) | FDs],[fila(D,Y,X,Ws1) | Fs]) :-` $\{\|FDs\|+1\}$

$\{true\}$

`1 < L,`

$\{1 < L\}$

$\{true\}$

`assoc_words(FDs,Fs),` $\{\|FDs\|\}$

$\{useful(\{\![FDs]\!\}) = \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![Fs]\!\} = \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\}\}$

$\{true\}$

`words(Ws),`

$\{\{\![Ws]\!\} = ws_0\}$

$\{true\}$

`words_of_ln(Ws,L,Ws1).`

$\{\{\![Ws1]\!\} = \{w \in \{\![Ws]\!\} \mid \|w\| = L\}\}$

$\{useful(\{\![[\texttt{filadim}(D,Y,X,L)|FDs]]\!\}) = \{\texttt{filadim}(D,Y,X,L)\} \cup \{\texttt{filadim}(D_i,Y_i,X_i,L_i) \mid i=1..n\} \wedge \{\![[\texttt{fila}(D,Y,X,Ws1)|Fs]]\!\} = \{\texttt{fila}(D,Y,X,Ws1)\} \cup \{\texttt{fila}(D_i,Y_i,X_i,Ws_i) \mid i=1..n\} \wedge \{\![Ws_i]\!\} = \{w \in ws_0 \mid \|w\| = L_i\} \wedge \{\![Ws1]\!\} = \{w \in ws_0 \mid \|w\| = L\}\}$

$\{true\}$

`ordina_res(Rs,Ws) :-`

$\{true\}$

`qs_file(Rs,Rs1),`

$\{\{\![Rs]\!\} = \{R_1,\ldots,R_n\} \wedge R_i <_r R_{i+1} \wedge Rs1 = [R_1,\ldots,R_n]\}$

$\{true\}$

`only_words(Rs1,Ws).`

$\{Rs1 = [R_1,\ldots,R_n] \wedge R_i = \texttt{res}(D_i,Y_i,X_i,W_i) \wedge Ws = [W_1,\ldots,W_n]\}$

$\{\{\![Rs]\!\} = \{R_1,\ldots,R_n\} \wedge R_i <_r R_{i+1} \wedge R_i = \texttt{res}(D_i,Y_i,X_i,W_i) \wedge Ws = [W_1,\ldots,W_n]\}$

$\{true\}$

`only_words([],[]).` $\{0\}$

$\{\texttt{[]} = \texttt{[]}\}$

$\{true\}$
`only_words([res(D,Y,X,W) | Rs],[W | Ws]) :-` $\{||Rs||+1\}$
$\{true\}$
`only_words(Rs,Ws).` $\{||Rs||\}$
$\{Rs = [R_1, \ldots, R_n] \wedge R_i = \mathtt{res}(D_i, Y_i, X_i, W_i) \wedge Ws = [W_1, \ldots, W_n]\}$
$\{[\mathtt{res}(D, Y, X, W)|Rs] = [\mathtt{res}(D, Y, X, W), R_1, \ldots, R_n] \wedge R_i = \mathtt{res}(D_i, Y_i, X_i, W_i) \wedge$
$[W|Ws] = [W, W_1, \ldots, W_n]\}$

$\{consistent(\{\![C]\!\}) \wedge compatible(\emptyset, \{\![C]\!\})\}$
`build_cross([],C,[]).` $\{0\}$
$\{consistent(\{\![C]\!\}) \wedge compatible(\emptyset, \{\![C]\!\}) \wedge \emptyset \in sol(\{\![C]\!\}, \emptyset)\}$

$\{consistent(\{\![C]\!\}) \wedge compatible(\{F\} \cup \{\![Fs]\!\}, \{\![C]\!\}\}$
`build_cross([F | Fs], C, [res(D,Y,X,W) | Rs]) :-` $\{||Fs||+1\}$
$\{true\}$
`min_fila([F | Fs], fila(D,Y,X,Ws), Fs1),`
$\{\{F\} \cup \{\![Fs]\!\} = \{\mathtt{fila}(D, Y, X, Ws)\} \cup \{\![Fs1]\!\}\}$
$\{true\}$
`member(W, Ws),`
$\{W \in \{\![Ws]\!\}\}$
$\{true\}$
`assegna(D,Y,X,W,C,C1),`
$\{\{\![C1]\!\} = assign(\{\![C]\!\}, W, D, Y, X)\}$
$\{consistent(\{\![C1]\!\})\}$
`aggiorna(Fs1,W,C1,Fs2),`
$\{consistent(\{\![C1]\!\}) \wedge \{\![Fs1]\!\} = \{F_1, \ldots, F_n\} \wedge \{\![Fs2]\!\} = \{F'_1, \ldots F'_n\} \wedge$
$F_i = \mathtt{fila}(D_i, Y_i, X_i, Ws_i) \wedge F'_i = \mathtt{fila}(D_i, Y_i, X_i, Ws'_i) \wedge$
$\{\![Ws'_i]\!\} = filter(D_i, Y_i, X_i, \{\![Ws_i]\!\} - \{W\}, C1)\}$
$\{consistent(\{\![C1]\!\}) \wedge compatible(\{\![Fs2]\!\}, \{\![C1]\!\})\}$
`build_cross(Fs2,C1,Rs).` $\{||Fs2||\}$
$\{consistent(\{\![C1]\!\}) \wedge compatible(\{\![Fs2]\!\}, \{\![C1]\!\}) \wedge \{\![Rs]\!\} \in sol(\{\![C1]\!\}, \{\![Fs2]\!\})\}$
$\{consistent(\{\![C]\!\}) \wedge compatible(\{F\} \cup \{\![Fs]\!\}, \{\![C]\!\}) \wedge$
$\{\mathtt{res}(D, Y, X, W)\} \cup \{\![Rs]\!\} \in sol(\{\![C]\!\}, \{F\} \cup \{\![Fs]\!\})\}$

The following lemmas have been used in this proof outline.

Lemma 4 *Let SC be a set of Cell, $D \in Dir$, $Y, X \in GaeN$ and $Ws = [W_1, \ldots, W_n] \in AtomList$. If $consistent(SC)$ and $compatible(\{\mathtt{fila}(D, Y, X, Ws)\}, SC)$ then $consistent(assign(SC, W_i, D, Y, X))$ for $i = 1..n$.*

Lemma 5 *Let $Ws \in AtomList$, $D \in Dir$, $Y, X \in GaeN$, SW be a set of Atom and SC be a set of Cell. Then:*

$$\{\![Ws]\!\} = filter(D, Y, X, SW, SC) \Longrightarrow compatible(\{\mathtt{fila}(D, Y, X, Ws)\}, SC).$$

Notice that $compatible(SF, SC)$ iff $compatible(\{F\}, SC)$ for each $F \in SF$.

$\{true\}$
`assegna(D,Y,X,W,C,C1) :-`
$\{true\}$
`atom_chars(W,W1),`
$\{W = W1\}$
$\{true\}$
`assegna2(D,Y,X,W1,C,C1).`

$\{\{\![C1]\!\} = assign(\{\![C]\!\}, W1, D, Y, X)\}$

$\{\{\![C1]\!\} = assign(\{\![C]\!\}, W, D, Y, X)\}$

$\{true\}$

`assegna2(D,Y,X,[],C,C).` $\{0\}$

$\{\{\![C]\!\} = assign(\{\![C]\!\}, \texttt{[]}, D, Y, X)\}$

$\{true\}$

`assegna2(orz,Y,X,[Ch|W],C,[cell(Y,X,Ch) | C1]) :-` $\{\|W\| + 1\}$

$\{true\}$

`X1 is X+1,`

$\{X1 = X + 1\}$

$\{true\}$

`assegna2(orz,Y,X1,W,C,C1).` $\{\|W\|\}$

$\{\{\![C1]\!\} = assign(\{\![C]\!\}, W, \texttt{orz}, Y, X + 1)\}$

$\{\{\texttt{cell}(Y, X, Ch)\} \cup \{\![C1]\!\} = assign(\{\![C]\!\}, [Ch|W], \texttt{orz}, Y, X)\}$

$\{consistent(\{\![C]\!\})\}$

`aggiorna([],W,C,[]).` $\{0\}$

$\{consisten(\{\![C]\!\}) \wedge \{\![\texttt{[]}]\!\} = \emptyset\}$

$\{consistent(\{\![C]\!\})\}$

`aggiorna([fila(D,Y,X,Ws)|Fs1],W,C,[fila(D,Y,X,Ws2)|Fs2]) :-` $\{\|Fs1\| + 1\}$

$\{true\}$

`minus_el(Ws,W,Ws1),`

$\{\{\![Ws1]\!\} = \{\![Ws]\!\} - \{W\}\}$

$\{consistent(\{\![C]\!\})\}$

`filtra_cross(D,Y,X,Ws1,C,Ws2),`

$\{consistent(\{\![C]\!\}) \wedge \{\![Ws2]\!\} = filter(D, Y, X, \{\![Ws1]\!\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\})\}$

`aggiorna(Fs1,W,C,Fs2).` $\{\|Fs1\|\}$

$\{consistent(\{\![C]\!\}) \wedge \{\![Fs1]\!\} = \{F_1, \ldots, F_n\} \wedge \{\![Fs2]\!\} = \{F'_1, \ldots F'_n\} \wedge$
$F_i = \texttt{fila}(D_i, Y_i, X_i, Ws_i) \wedge F'_i = \texttt{fila}(D_i, Y_i, X_i, Ws'_i) \wedge$
$\{\![Ws'_i]\!\} = filter(D_i, Y_i, X_i, \{\![Ws_i]\!\} - \{W\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\}) \wedge \{\![[\texttt{fila}(D, Y, X, Ws)|Fs1]]\!\} = \{\texttt{fila}(D, Y, X, Ws), F_1, \ldots, F_n\} \wedge$
$\{\![[\texttt{fila}(D, Y, X, Ws2)|Fs2]]\!\} = \{\texttt{fila}(D, Y, X, Ws2), F'_1, \ldots F'_n\} \wedge$
$F_i = \texttt{fila}(D_i, Y_i, X_i, Ws_i) \wedge F'_i = \texttt{fila}(D_i, Y_i, X_i, Ws'_i) \wedge$
$\{\![Ws'_i]\!\} = filter(D_i, Y_i, X_i, \{\![Ws_i]\!\} - \{W\}, \{\![C]\!\}) \wedge$
$\{\![Ws2]\!\} = filter(D, Y, X, \{\![Ws]\!\} - \{W\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\})\}$

`filtra_cross(D,Y,X,[],C,[]).` $\{0\}$

$\{consistent(C) \wedge filter(D, Y, X, \emptyset, \{\![C]\!\}) = \emptyset\}$

$\{consistent(\{\![C]\!\})\}$

`filtra_cross(D,Y,X,[W|Ws],C,[W|Ws1]) :-` $\{\|Ws\| + 1\}$

$\{true\}$

`atom_chars(W,W1),`

$\{W = W1\}$

$\{consistent(\{\![C]\!\})\}$

`filtra_cross_aux(D,Y,X,W1,C),`

$\{consistent(\{\![C]\!\}) \wedge filter(D, Y, X, \{W1\}, \{\![C]\!\}) = \{W1\}\}$

$\{consistent(\{\![C]\!\})\}$

```
    filtra_cross(D,Y,X,Ws,C,Ws1).
```
$\{\|Ws\|\}$

$\{consistent(\{\![C]\!\}) \land \{\![Ws1]\!\} = filter(D, Y, X, \{\![Ws]\!\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\}) \land \{\![[W|Ws1]]\!\} = filter(D, Y, X, \{\![[W|Ws]]\!\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\})\}$

```
filtra_cross(D,Y,X,[W|Ws],C,Ws1) :-
```
$\{\|Ws\| + 1\}$

$\{true\}$

```
    atom_chars(W,W1),
```
$\{W1 = W\}$

$\{consistent(\{\![C]\!\})\}$

```
    \+ filtra_cross_aux(D,Y,X,W1,C),
```
$\{consistent(\{\![C]\!\}) \land filter(D, Y, X, \{W1\}, \{\![C]\!\}) = \emptyset\}$

$\{consistent(\{\![C]\!\})\}$

```
    filtra_cross(D,Y,X,Ws,C,Ws1).
```
$\{\|Ws\|\}$

$\{consistent(\{\![C]\!\}) \land \{\![Ws1]\!\} = filter(D, Y, X, \{\![Ws]\!\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\}) \land \{\![Ws1]\!\} = filter(D, Y, X, \{\![[W|Ws]]\!\}, \{\![C]\!\})\}$

$\{consistent(\{\![C]\!\})\}$

```
filtra_cross_aux(D,Y,X,[],C).
```
$\{0\}$

$\{consistent(\{\![C]\!\}) \land filter(D, Y, X, \{\texttt{[]}\}, \{\![C]\!\}) = \{\texttt{[]}\}\}$

$\{consistent(\{\![C]\!\})\}$

```
filtra_cross_aux(orz,Y,X,[Ch|W],C) :-
```
$\{\|W\| + 1\}$

$\{consistent(\{\![C]\!\})\}$

```
    possibly_in(Y,X,Ch,C),
```
$\{consistent(\{\![C]\!\}) \land \texttt{cell}(Y, X, Ch1) \in \{\![C]\!\} \Longrightarrow Ch1 = Ch\}$

$\{true\}$

```
    X1 is X+1,
```
$\{X1 = X + 1\}$

$\{consistent(\{\![C]\!\})\}$

```
    filtra_cross_aux(orz,Y,X1,W,C).
```
$\{\|W\|\}$

$\{consistent(\{\![C]\!\}) \land filter(\texttt{orz}, Y, X + 1, \{W\}, \{\![C]\!\}) = \{W\}\}$

$\{consistent(\{\![C]\!\}) \land filter(\texttt{orz}, Y, X, \{[Ch|W]\}, \{\![C]\!\}) = \{[Ch|W]\}\}$

Notice that both $filter(\texttt{orz}, Y, X+1, \{W\}, SC) = \{W\}$ and $\texttt{cell}(Y, X, Ch1) \in SC \Longrightarrow Ch1 = Ch$ hold if and only if $filter(\texttt{orz}, Y, X, \{[Ch|W]\}, SC) = \{[Ch|W]\}$.

$\{consistent(\texttt{cell}(Y, X, Ch) \cup \{\![C]\!\})\}$

```
possibly_in(Y,X,Ch,[cell(Y,X,Ch)|C]).
```
$\{\|C\| + 1\}$

$\{consistent(\texttt{cell}(Y, X, Ch) \cup \{\![C]\!\}) \land$
$\texttt{cell}(Y, X, Ch1) \in \{\texttt{cell}(Y, X, Ch)\} \cup \{\![C]\!\} \Longrightarrow Ch1 = Ch\}$

$\{consistent(\emptyset)\}$

```
possibly_in(Y,X,Ch,[]).
```
$\{0\}$

$\{consistent(\emptyset)\}$

$\{consistent(\{\texttt{cell}(Y1, X1, Ch1)\} \cup \{\![C]\!\})\}$

```
possibly_in(Y,X,Ch,[cell(Y1,X1,Ch1)|C]) :-
```
$\{\|C\| + 1\}$

$\{true\}$

```
    Y =\= Y1,
```
$\{Y \neq Y1\}$

$\{consistent(\{\![C]\!\})\}$

`possibly_in(Y,X,Ch,C).` $\{\|C\|\}$

$\{consistent(\{\!\!|C|\!\!\}) \wedge \texttt{cell}(Y,X,Ch') \in \{\!\!|C|\!\!\} \Longrightarrow Ch' = Ch\}$

$\{consistent(\{\texttt{cell}(Y1,X1,Ch1)\} \cup \{\!\!|C|\!\!\}) \wedge$
$\texttt{cell}(Y,X,Ch') \in \{\texttt{cell}(Y1,X1,Ch1)\} \cup \{\!\!|C|\!\!\} \Longrightarrow Ch' = Ch\}$

Notice that, if $consistent(SC)$ and $\texttt{cell}(Y,X,Ch) \in SC$ then $\texttt{cell}(Y,X,Ch') \in SC \Longrightarrow Ch' = Ch$.

$\{true\}$
`min_fila([F],F,[]).` $\{1\}$
$\{\{F\} = \{F\}\}$

$\{true\}$
`min_fila([F|Fs],M,[F|Fs1]) :-` $\{\|Fs\| + 1\}$
$\{true\}$
`min_fila(Fs,M,Fs1),` $\{\|Fs\|\}$
$\{\{\!\!|Fs|\!\!\} = \{M\} \cup \{\!\!|Fs1|\!\!\}\}$
$\{true\}$
`shorter_fila(M,F).`
$\{M = \texttt{fila}(D,Y,X,Ws) \wedge F = \texttt{fila}(D',Y',X',Ws') \wedge \|Ws\| \leq \|Ws'\|\}$
$\{\{\!\!|Fs|\!\!\} \cup \{F\} = \{M,F\} \cup \{\!\!|Fs1|\!\!\}\}$

$\{true\}$
`min_fila([F|Fs],F,[M|Fs1]) :-` $\{\|Fs\| + 1\}$
$\{true\}$
`min_fila(Fs,M,Fs1),` $\{\|Fs\|\}$
$\{\{\!\!|Fs|\!\!\} = \{M\} \cup \{\!\!|Fs1|\!\!\}\}$
$\{true\}$
`\+ shorter_fila(M,F).`
$\{M = \texttt{fila}(D,Y,X,Ws) \wedge F = \texttt{fila}(D',Y',X',Ws') \wedge \|Ws\| > \|Ws'\|\}$
$\{\{\!\!|Fs|\!\!\} \cup \{F\} = \{M,F\} \cup \{\!\!|Fs1|\!\!\}\}$

Theorem 6

- *The program is weakly partial correct.*
- *The program terminates.*

4 About (strong) correctness

The program is also (strongly) totally correct. In fact, each solution of the puzzle is found by the program. However, to prove this we should exhibit a model supported by the program. There are some difficulties, which arise since in the program we extensively use lists to represent (multi)sets. A set can be represented by several lists which differ only in the order of the elements and during the esecution of the program just one of these lists is generated to represent a particular set. To have a supported model, we should exactly specify the representative list, whereas the intended semantics specifies only the corresponding set.

For instance, consider the predicate `minmax`, that, given a set of integer, returns its minimum element, its maximum element and the rest of the set:

```
minmax([M,N],M,N,[]) :-
    M =< N.
minmax([N,M],M,N,[]) :-
```

```
    N > M.
minmax([L|Ns],M,N,[L|Ns1]) :-
    minmax(Ns,M,N,Ns1),
    M =< L, L =< N.
minmax([L|Ns],L,N,[M|Ns1]) :-
    minmax(Ns,M,N,Ns1),
    L < M.
minmax([L|Ns],M,L,[N|Ns1]) :-
    minmax(Ns,M,N,Ns1),
    L > N.
```

The intended semantics is given by the pre- and post-conditions

$$Pre = \{\mathtt{minmax}(Ns, M, N, Ns1) \mid Ns \in GaeNList \wedge \|Ns\| \geq 2\}$$

$$Post = \{\mathtt{minmax}(Ns, M, N, Ns1) \mid \{\!|Ns|\!\} = \{\!|Ns1|\!\} \cup \{M, N\} \wedge M = \min\{\!|Ns|\!\} \wedge N = \max\{\!|Ns|\!\}\}$$

However, to obtain a supported model, we should specify *exactly* how list $Ns1$ is obtained from Ns. This is not easy: for instance

$$\mathtt{minmax}([1, 8, 7, 2, 6, 5, 4, 3], M, N, Ns1)$$

returns:

$$M = 1, \quad N = 8, \quad Ns1 = [2, 7, 6, 3, 5, 4]$$

In our opinion, the "correct" post-condition does not respect the intended semantics, since it specifies uninteresting implemention details.

This particular procedure could be easily changed. However, lists are extensively used in our program to represent sets, and it would be very boring to define standard representatives for all those sets.

As a general issue, it would be important to have techniques to deal with abstract data types: in this case it would possible to consider lists which represent sets just as "sets" (and to avoid the heavy usage of $\{\!|List|\!\}$).

The necessity to give a supported model is a too strong condition also from another point of view: in proving correctness, in fact, it is useful to abstract away from implementation details, that can improve efficiency, but that are not meaningful for correctnes.

For instance, in the definition of the post-condition of $\mathtt{min\_fila}(Fs, F, Fs1)$ we do not care about the fact that F is the smallest $Fila$ of Fs. For `build_cross` to be correct, in fact, it is only important that F is an element of Fs. The choice of the minimal element is important only to have a more efficient program.

5 Parallelizability via delay declarations

The program is intrinsecally sequential. Some clauses could however be parallelized, in particular those which map a function on a set (`assoc_words`, `aggiorna`) or which filter a set (`filtra_cross`), or the routines like `qs` which have already be analized in the literature. The techniques we know to deal with parallelizability via delay declaretions cannot be applied to the program.