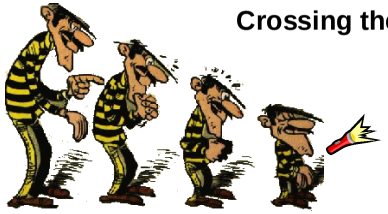
KBS
Knowledge-Based Systems Group

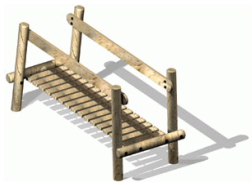
Answer Set Planning with DLV^K

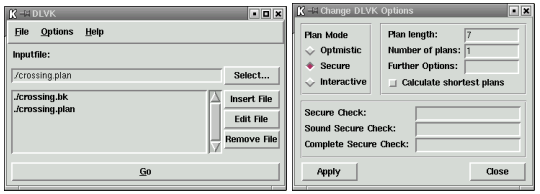
Planning with Action Costs

TU WIEN
TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

Crossing the Bridge





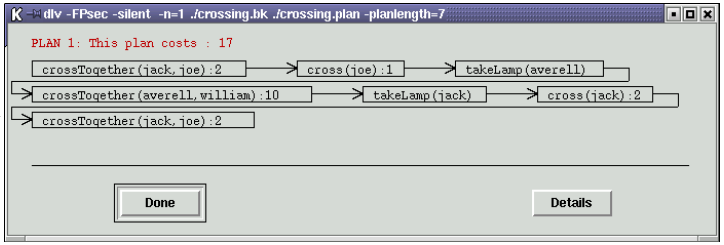


Planning Language K

```
person(joe).
person(jack).
person(william).
person(averell).
speed(joe,1).
speed(jack,2).
speed(william,5).
speed(averell,10).

max(A,B,A) :- A >=B, speed(_,A), speed(_,B).
max(A,B,B) :- B > A, speed(_,A), speed(_,B).

actions:
    crossTogether(X,Y) requires person(X), person(Y), X < Y
                       costs Smax where speed(X,SX), speed(Y,SY),
                                        max(SX,SY,Smax).
    cross(X) requires person(X) costs SX where speed(X,SX).
    takeLamp(X) requires person(X).

fluents:
    across(X) requires person(X).
    hasLamp(X) requires person(X).
    differentSides(X,Y) requires person(X), person(Y).

initially:
    -across(X).
    hasLamp(joe).

always:
    executable crossTogether(X,Y) if hasLamp(X).
    executable crossTogether(X,Y) if hasLamp(Y).
    nonexecutable crossTogether(X,Y) if differentSides(X,Y).

    executable cross(X) if hasLamp(X).

    executable takeLamp(X).
    nonexecutable takeLamp(X) if hasLamp(Y), differentSides(X,Y).

    caused across(X) after crossTogether(X,Y). -across(X).
    caused across(Y) after crossTogether(X,Y). -across(Y).
    caused -across(X) after crossTogether(X,Y). across(X).
    caused -across(Y) after crossTogether(X,Y). across(Y).

    caused across(X) after cross(X). -across(X).
    caused -across(X) after cross(X). across(X).

    caused hasLamp(X) after takeLamp(X).
    caused -hasLamp(X) after takeLamp(Y), X !=Y, hasLamp(X).

    caused differentSides(X,Y) if across(X). -across(Y).
    caused differentSides(X,Y) if -across(X). across(Y).

    inertial across(X).
    inertial -across(X).
    inertial hasLamp(X).

noConcurrency.

goal: across(joe). across(jack). across(william). across(averell)?
```

DLV^K Planning Frontend

Plans

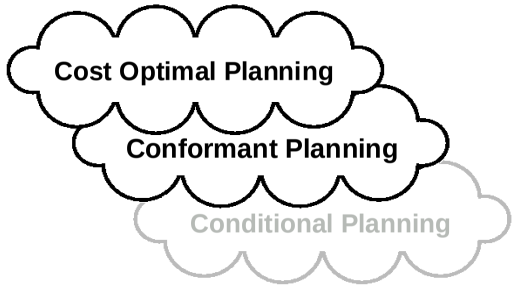


Disjunctive Logic Program

```
crossTogether(X,Y,T) v -crossTogether(X,Y,T) :- hasLamp(X,T),actiontime(T),
                                                person(X), person(Y), X < Y.
crossTogether(X,Y,T) v -crossTogether(X,Y,T) :- hasLamp(Y,T),actiontime(T),
                                                person(X), person(Y), X < Y,
:- differentSides(X,Y,T), crossTogether(X,Y,T), actiontime(T).

costs_crossTogether(X,Y,T,C) :- person(X), person(Y), X < Y,
                                speed(X,SX), speed(Y,SY),
                                max(SX,SY,C); crossTogether(X,Y,T).
:~ costs_crossTogether(X,Y,T,C). [ C, C]

cross(X,T) v -cross(X,T) :- hasLamp(X,T), actiontime(T), person(X).

costs_cross(X,T,SX) :- person(X), speed(X,SX), cross(X,T).
:~ costs_cross(X,T,C). [ C, C]

takeLamp(X,T) v -takeLamp(X,T) :- #actiontime(T), person(X).

-across(X,0) :- person(X).
hasLamp(joe,0) :- person(joe).

:- hasLamp(Y,T), differentSides(X,Y,T), takeLamp(X,T), actiontime(T).

across(X,T1)  :- crossTogether(X,Y,T), -across(X,T),
                 nexttime(T,T1), person(X).
across(Y,T1)  :- crossTogether(X,Y,T), -across(Y,T),
                 nexttime(T,T1), person(Y).
-across(X,T1) :- crossTogether(X,Y,T), across(Y,T),
                 nexttime(T,T1), person(X).
-across(Y,T1) :- crossTogether(X,Y,T), across(Y,T),
                 nexttime(T,T1), person(Y).

across(X,T1)  :- cross(X,T), -across(X,T), nexttime(T,T1), person(X).
-across(X,T1) :- cross(X,T), across(X,T), nexttime(T,T1), person(X).

hasLamp(X,T1)  :- takeLamp(X,T), nexttime(T,T1), person(X).
-hasLamp(X,T1) :- takeLamp(Y,T), X != Y, hasLamp(X,T),
                  nexttime(T,T1), person(X).

differentSides(X,Y,T) :- across(X,T), -across(Y,T), time(T),
                         person(X), person(Y).
differentSides(X,Y,T) :- -across(X,T), across(Y,T), time(T),
                         person(X), person(Y).

across(X,T1)  :- across(X,T), nexttime(T,T1), person(X), not -across(X,T1).
-across(X,T1) :- -across(X,T), nexttime(T,T1), person(X), not across(X,T1).
hasLamp(X,T1) :- hasLamp(X,T1), nexttime(T,T1), person(X), not -hasLamp(X,T1).

:- crossTogether(X,Y,T), crossTogether(X1,Y1,T), Y != Y1.
:- crossTogether(X,Y,T), crossTogether(X1,Y1,T), X != X1.
:- crossTogether(X,Y,T), cross(X1,T).
:- crossTogether(X,Y,T), takeLamp(X1,T).
:- cross(X,T), cross(X1,T), X != X1.
:- cross(X,T), takeLamp(X1,T).
:- takeLamp(X,T), takeLamp(X1,T), X != X1.

goal :- across(joe,0), across(jack,0), across(willian,0), across(averell,0).
:- not goal.
```

DLV Answer Set Engine

Answer Sets



Project Team:
Thomas Eiter (TU Wien)
Wolfgang Faber (TU Wien)
Nicola Leone (University of Calabria)
Gerald Pfeifer (TU Wien)
Axel Polleres (TU Wien)

Contact Information:
Axel Polleres
Institute of Information Systems
Knowledge-Based Systems Group 184/
Favoritenstrasse 9-11
A-1040 Vienna
Phone: +43 1 58801-18462
Email: *axel@kr.tuwien.ac.at*

This work is supported by FWF (Austrian Science Funds) under the pojects P14781 and Z29-INF.