Domain-Dependent Knowledge in Answer Set Planning
The Block World Experiment

Tran Cao Son* Chitta Baral† and Tran Hoai Nam† Sheila McIlraith‡

*Computer Science
New Mexico State University
PO Box 30001, MSC CS
Las Cruces, NM 88003, USA
tson@cs.nmsu.edu

†Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA
{chitta,namtran}@asu.edu

‡Computer Science
Knowledge Systems Laboratory
Stanford University
Stanford, CA 94305
sam@ksl.stanford.edu

December 11, 2003

In the block world domain, given stacks of blocks on an infinite table, we need to find plans for a robot hand to move the blocks into a desired arrangement. The fluents in the domain and their meanings are described as follows.

- $on(X, Y)$ - block X is on block Y.
- $on(X, table)$ - block X is on the table.
- $clear(X)$ - block X is clear, that is, there is no block on X.
- $holding(X)$ - block X is being held in the robot hand.

There are three actions in the domain as follows.

- $take(X)$ - block X is picked up.
- $placeOn(Y)$ - the block held in the robot hand is put on block Y.
- $placeOn(table)$ - the block held in the robot hand is put on the table.

The domain description includes the following propositions.

$$D_{Block} = \left\{ \begin{array}{l} \textbf{causes}(take(X), \neg handempty, \{\}) \\ \textbf{causes}(take(X), \neg clear(X), \{\}) \\ \textbf{causes}(take(X), holding(X), \{\}) \\ \textbf{causes}(take(X), clear(Y), \{on(X,Y)\}) \\ \textbf{causes}(take(X), \neg on(X,Y), \{on(X,Y)\}) \\ \textbf{causes}(placeOn(Y), handempty, \{\}) \\ \textbf{causes}(placeOn(Y), clear(X), \{holding(X)\}) \\ \textbf{causes}(placeOn(Y), \neg holding(X), \{holding(X)\}) \\ \textbf{causes}(placeOn(Y), \neg clear(Y), \{\}) \\ \textbf{causes}(placeOn(Y), on(X,Y), \{holding(X)\}) \\ \textbf{executable}(take(X), \{clear(X), handempty\}) \\ \textbf{executable}(placeOn(Y), \{clear(Y), \neg handempty\}) \end{array} \right.$$

In the above propositions, X denotes a block, Y denotes a block or the table. Again, a proposition with variables denotes the set of its ground instances. Let G is the set of goal block positions, that is, $G = \{on(X,Y) | X \text{ is on } Y \text{ in the goal state }\}$. We define the following procedural control knowledge for the blocks world domain:

$$S = \begin{cases} (control & : & \textbf{while}\ \ (\neg goal)\ \textbf{do}\ \ (build|remove)) \\ (build & : & \textbf{pick}((X,Y), Block^2, buid\_good\_tower(X,Y)) \\ (remove & : & \textbf{pick}(Z, Block, remove\_bad\_block(Z)) \\ (build\_good\_tower(X,Y) & : & \textbf{if}\ \ (good\_tower(Y) \wedge \neg on(X,Y) \wedge clear(X)) \\ & & \quad \textbf{then}\ \ (take(X); placeOn(Y)))\ \textbf{else}\ \ \textbf{null} \\ (remove\_bad\_block(Z) & : & \textbf{if}\ \ (to\_be\_clear(Y) \wedge on\_top(Z,Y)) \\ & & \quad \textbf{then}\ \ (take(Z); placeOn(table)))\ \textbf{else}\ \ \textbf{null} \end{cases}$$

where $Block$ denotes the set of all block constants and $Block^2$ denotes the set of all pairs of block constants and $goal$, $good\_tower(X)$, $to\_be\_clear(Y)$, $on\_top(Z,Y)$ are formula names. The formulae associated to them are defined below[1].

$$\begin{aligned} goal &\stackrel{\text{def}}{=} \bigwedge\nolimits_{on(X,Y)\in G} on(X,Y) \\ good\_tower(X) &\stackrel{\text{def}}{=} ((on(X,table) \in G) \wedge on(X,table)) \\ &\quad\ \bigvee(\exists Y : (on(X,Y) \in G) \wedge on(X,Y) \wedge good\_tower(Y)) \\ to\_be\_clear(X) &\stackrel{\text{def}}{=} right\_place\_but\_blocked(X) \bigvee to\_move\_but\_blocked(X) \\ &\quad\ \bigvee to\_move\_onto\_table(X) \\ right\_place\_but\_blocked(X) &\stackrel{\text{def}}{=} good\_tower(X) \wedge \neg clear(X) \bigwedge(\forall Y : on(Y,X) \in G \Rightarrow \neg on(Y,X)) \\ to\_move\_but\_blocked(X) &\stackrel{\text{def}}{=} (\exists Y : (on(X,Y) \in G) \wedge good\_tower(Y) \wedge clear(Y)) \\ &\quad\ \bigwedge \neg clear(X) \bigwedge \neg holding(X) \\ to\_move\_onto\_table(X) &\stackrel{\text{def}}{=} (on(X,table) \in G) \bigwedge \neg on(X,table) \bigwedge \neg clear(X) \bigwedge \neg holding(X) \\ on\_top(X,Y) &\stackrel{\text{def}}{=} above(X,Y) \bigwedge clear(X) \\ above(X,Y) &\stackrel{\text{def}}{=} on(X,Y) \bigvee(\exists Z : block(Z) \wedge above(X,Z), above(Z,Y)) \end{aligned}$$

Intuitively, $goal$ becomes true once all the blocks are in their goal positions; $good\_tower(X)$ is true when X and all the blocks under X (in the same tower) are in the goal positions; $to\_be\_clear(X)$ says that block X must become clear; $right\_place\_but\_blocked(X)$ means the block on X is blocking some block Y from achieving the goal position $on(Y,X)$; $to\_move\_but\_blocked(X)$ (or $to\_move\_onto\_table(X)$) says that X can not move to its goal position $on(X,Y)$ (or $on(X,table)$) although Y (or the table) is clear; $on\_top(X,Y)$ states that block X is the top block of the tower containing Y; $above(X,Y)$ is true if X and Y are in the same tower where X lies above Y.

Again, we run the program with and without the control knowledge. We ran our experiment under Windows XP on a desktop with 256Mb RAM and an Intel Celeron 2.2 GHz processor, using **lparse** version 1.0.4 (Windows, build Apr 5, 2001) and **smodels** version 2.26. The results are described in the following table.

[1] For easy of reading, we write the formulae using the conventional operators such as $\wedge, \vee, \Rightarrow$ etc. The left hand side of an expression "$. \stackrel{\text{def}}{=} .$" is the name assigned to the formula on the right hand side of the expression.

Problem	With Control	Knowledge	Without Control	Knowledge
	Length	Time	Length	Time
4-0	6	0.313	12	0.531
4-1	10	0.312	10	0.421
4-2	6	0.313	12	0.547
5-0	12	0.641	16	1.562
5-1	10	0.671	16	1.343
5-2	16	0.671	16	0.889
6-0	12	1.156	20	5.296
6-1	10	1.156	20	6.062
6-2	20	1.156	20	4.905
7-0	20	1.984	24	64.078
7-1	22	1.952	24	110.281
7-2	20	1.999	24	57.343
8-0	18	3.045	28	34.795
8-1	20	3.046	28	417.437
8-2	16	3.141	28	1060.515
9-0	30	4.578	n/a	n/a
9-1	28	4.64	n/a	n/a
9-2	26	4.499	n/a	n/a
10-2	34	6.578	n/a	n/a
11-0	32	9.141	n/a	n/a
11-1	30	9.313	n/a	n/a
11-2	34	9.217	n/a	n/a
12-0	34	12.202	n/a	n/a
12-1	34	12.921	n/a	n/a
13-0	42	17.578	n/a	n/a
13-1	44	17.313	n/a	n/a
14-0	38	21.343	n/a	n/a
14-1	36	22.421	n/a	n/a
15-0	40	27.265	n/a	n/a
15-1	52	28.547	n/a	n/a
16-1	54	36.843	n/a	n/a
16-2	52	36.39	n/a	n/a
17-0	46	45.171	n/a	n/a

We can see from the table that in the experiment, planning with control knowledge yield better time performance as well as plan quality. For each of the problems from number 9-0 to number 17-0, the smodels program did not return after 2 hours and we decided to abort in these case.