

# Errata for *Automated Planning: Theory and Practice*

Dana S. Nau, University of Maryland

May 4, 2005

**Page ii, 5th line from bottom.** Remove “for a list of publications.”

**Page xxiv, last paragraph.** replace this paragraph with the following:

The web site for the book can be found at <http://www.laas.fr/planning> or <http://books.elsevier.com/mk/1558608567>. It contains a complete set of lecture slides, and other auxiliary materials.

**Page 6, Example 1.1, 3rd line.** Change “can can” to “can”.

**Page 10, last paragraph.** Change A0 to A1.

**Page 11, 3rd line before Section 1.6.** Change II to III.

**Page 25, middle of the page.**  $i = 0, \dots, n$  should be  $i = 0, \dots, m$ .

**Pages 25 and 26.** All occurrences of “ $\text{val}_{i+1 \bmod m}$ ” should be  $\text{val}_{i+1 \bmod m}$ ”.

**Page 30, 2nd line of Example 2.10.** c1,c2 should be c3,c1.

**Page 31, last paragraph.** “ $g$  contains negated atoms” should be “ $g$  may contain negated atoms”, and “Definition 2.1 (see page 20)” should be “on page 22”.

**Page 32, Example 2.11, 8th line.** l1 should be loc1.

**Page 33, 2nd paragraph, 2nd line.** “is” should be “be”.

**Page 33, 2nd bullet.**  $t$  and  $I(t)$  should be  $s$  and  $I(s)$ .

**Page 35, last line.** Interchange the superscripts  $+$  and  $-$ .

**Page 48, 4th line.**  $x_n$  should be  $t_n$ .

**Page 51, 5th line.** “effects” should be “precond”.

**Page 51, 6th line.** “precond” should be “effects”.

**Page 56, 2nd bullet.**  $P$  should be  $(P, k)$ .

**Page 57.** Remove the paragraph before Proposition 3.1.

**Page 73, line 5 of Section 4.3.** The equation should be

$$\Gamma^{-1}(g) = \{\gamma^{-1}(g, a) \mid a \in A \text{ is relevant for } g\}$$

**Page 73, 4th line from bottom.** applicable should be relevant.

**Page 74, middle of page.** “ $r1$ ” should be “ $r1$ ”. One line earlier, “ $\text{at}(r1, \text{loc1})$ ,” should be removed. Three lines later, “ $\text{occupied}(\text{loc1})$ ” should be “ $\neg\text{occupied}(\text{loc1})$ ”.

**Page 79, last line before Section 4.5.2.** In “ $\text{position}(c1, s_0)$ ”,  $c1$  should be  $c3$ .

**Page 88, first line of Definition 5.1.** “set” should be “multiset”.

**Page 107, line 12 of Exercise 5.7.** Both occurrences of  $\text{status}(x, \text{fill})$  should be  $\text{status}(x, \text{filling})$ .

**Page 121, 4th line of Definition 6.4.** “actions” should be “actions, see p. 124”.

**Page 126, 7th line of Example 6.3.**  $\text{Mr}21$  should be  $\text{Mr}12$ .

**Page 157, line 12.** Interchange “function” and “cost”.

**Page 165, exercise 7.3.** After “robots”, insert “that can be in the same location at the same time”.

**Page 171, Examples 8.3 and 8.4.**  $\{(\alpha, \alpha), (\beta, \beta), (\gamma, \gamma)\}$  should be  $\{(\alpha, \alpha); (\beta, \beta); (\gamma, \gamma)\}$ .

**Page 171, 5th line from bottom.** “due” should be “dual”.

**Pages 174–177.** Replace Section 8.3.1 with the pages at the end of the errata list.

**Page 191.** Add the following:

**8.11** Modify the encoding of Frame Axioms (Step 4 in Section 8.3.1) such as to get only binary constraints (hint: add a new CSP variable for each action and for each variable invariant for that action).

**Page 202, 2nd and 3rd lines of (9.1).** Replace the two lines with

$$\begin{aligned}\Delta_0(s, g) &= 0 && \text{if } g \subseteq s \\ \Delta_0(s, g) &= 0 && \text{if } p \notin s \text{ and } \forall a \in A, p \notin \text{effects}^+(a)\end{aligned}$$

**Page 202, first line after (9.1).** Replace “an estimate” with “the exact value”.

**Page 203, Figure 9.2.** Replace “ $U \leftarrow \{s\}$ ” with “ $U \leftarrow s$ ”.

Replace “ $\exists u \in U, \text{precond}(a) \subseteq u$ ” with “ $\text{precond}(a) \subseteq U$ ”.

Replace “ $U \leftarrow \{u\} \cup \text{effects}^+(a)$ ” with “ $U \leftarrow U \cup \text{effects}^+(a)$ ”.

**Page 213, last paragraph, 6th line.** “were” should be “when”.

**Page 213, last paragraph, 8th line.** ’99 should be ’98.

**Page 220, Section 10.3, 2nd paragraph.** Remove the space after “ $\text{progr}$ ”.

**Page 240, Example 11.8, 3rd line.**  $\text{loc}2$  should be  $l2$ .

**Page 244, last paragraph, 3rd line.**  $\pi$  should be  $\pi = \langle a_1, \dots, a_n \rangle$ .

**Page 269, 5th paragraph, 2nd line.**  $\text{load}(r, c)$  should be  $\text{load}(r1, c1)$ .

**Page 274, 2nd through 12th lines.** Put space before (domain), (move l2), etc. to line them up in a column.

**Page 283, 1st bullet.** “location 1 to location 2” should be “location 2 to location 1”.

**Page 295, 5th and 8th lines of the text.** Replace all four occurrences of  $\cup$  with  $\bullet$ .

**Page 299, last line.** “a” should be “an”.

**Page 302, 3rd line.** Replace “As mentioned earlier,  $IA_c$  is” with “Let  $IA_c$  be”.

**Page 303, title of Section 13.4.1.** Replace “Constraints” with “Problems”.

**Page 303, Section 13.4.1, 3rd paragraph, 1st line.** Remove “constraint”.

**Page 311, 10th line before bottom.** Remove “for”.

**Page 322, 2nd and 3rd bullets.** Replace  $C$  with  $\theta$ .

**Page 338, 2nd line of the figure.** Replace “return( $\pi$ )” with “return( $\pi, \Phi$ )”.

**Page 346, Figure 14.8.** filling(bt), filling(wm), and filling(dw) should be fill(bt), fill(wm), and fill(dw).

**Page 371, Figure 15.9, 6th line.**  $\theta(\alpha/\Phi)$  should be  $\{\theta(\alpha/\Phi)\}$ .

**Page 393, 6th-8th lines from bottom.** replace f and e with full and empty throughout.

**Page 395, 5th line from bottom.** “function” should be “partial function”.

**Page 396, 3rd line.**  $b(s3) = 0$  should be  $b(s4) = 0$ .

**Page 396, 5th line.** observe – container should be observe-container.

**Page 403, end of 1st paragraph.** Replace “conventions” with “ideas”.

**Page 415, 3rd and 4th bullets.**  $S \times C \times A$  should be  $S \times C \rightarrow A$ , and  $S \times C \times S \times C$  should be  $S \times C \times S \rightarrow C$ .

**Page 421, Section 17.3.3.** Replace “In spite of ... seem to be” with “Temporal logics cannot express goals that are”.

**Page 432, 1st paragraph, 4th line.** Replace “a satisfiability problem” with “the problem of finding a plan that satisfies the specification”.

*The next four pages contain the replacement for Section 8.3.1.*

### 8.3.1 Encoding a Planning Problem into a CSP

A bounded planning problem  $P = (O, R, s_0, g, k)$  in the state variable representation is encoded into a constraint satisfaction problem  $P'$  in four steps corresponding respectively to (1) the definition of the CSP variables of  $P'$ , (2) the definition of the constraints encoding the initial state  $s_0$  and the goal  $g$ , (3) the encoding of the actions that are instances of operators in  $O$ , and (4) the encoding of the frame axioms.

The set of solutions of the CSP  $P'$  is intended to correspond to the set of plans  $\langle a_1, \dots, a_k \rangle$  of  $P$  of length  $\leq k$ . We are interested in characterizing the sequences of states  $\langle s_0, s_1, \dots, s_k \rangle$  corresponding to such plans. For convenience, let us refer to the state  $s_j$  in this sequence by its index  $j$ , for  $0 \leq j \leq k$ .

**Step 1: CSP variables.** The CSP variables of  $P'$  are defined as follows:

- for each ground state variable  $x_i$  of  $P$  ranging over  $D_i$  and for  $0 \leq j \leq k$ , there is a CSP variable of  $P'$ ,  $x_i(j, v_u, \dots, v_w)$  whose domain is  $D_i$ ,
- for  $0 \leq j \leq k - 1$ , there is a CSP variable, denoted  $\text{act}(j)$ , whose domain is the set of all possible actions, in addition to a `noop` action that has no preconditions and no effects, i.e.,  $\forall s, \gamma(s, \text{noop}) = s$ . More formally:

$$\text{act} : \{0, \dots, k - 1\} \rightarrow D_{\text{act}}, \text{ and}$$

$$D_{\text{act}} = \{a(v_u, \dots, v_w) \text{ ground instance of } o \in O\} \cup \{\text{noop}\}$$

Hence, the CSP variables are all the ground state variables of  $P$ , plus one variable  $\text{act}(j)$  whose value corresponds to the action carried out in state  $j$ .

**Example 8.14** Let  $P = (O, R, s_0, g)$ , where  $O$  are the operators given in Example 8.13 for a simplified DWR domain with one robot `r1`, one container `c1`, and two adjacent locations `l1`, `l2`. Since there is a single robot and a single container, let us simplify the notation by removing the arguments  $r$  and  $c$  in the variables `rloc`, `rload`, `cpos`, and in the operators. Let  $s_0$  be the following state  $s_0 = \{\text{rloc}=\text{l1}, \text{rload}=\text{nil}, \text{cpos}=\text{l2}\}$ ; and let  $g = \{\text{cpos}=\text{l1}\}$ . A plan for this problem consists in moving the robot from `l1` to `l2`, loading the container, moving back to `l1` and unloading. Assume that we are seeking a plan of at most  $k = 4$  steps. The corresponding CSP  $P'$  has the following set of variables:

- $\text{rloc}(j) \in \{\text{l1}, \text{l2}\}$ , for  $0 \leq j \leq 4$ ;
- $\text{rload}(j) \in \{\text{c1}, \text{nil}\}$ , for  $0 \leq j \leq 4$ ;
- $\text{cpos}(j) \in \{\text{l1}, \text{l2}, \text{r1}\}$ , for  $0 \leq j \leq 4$ ;

- $\text{act}(j) \in \{\text{move}(l, m), \text{load}(l), \text{unload}(l), \text{noop}\}$ , for all the possible instances of these operators and for  $0 \leq j \leq 3$ .

In this problem there are  $6 \times 5 - 1 = 29$  CSP variables.  $\square$

**Step 2: Constraints encoding  $s_0$  and  $g$ .** The encoding of the state  $s_0$  and the goal  $g$  into constraints follows directly from the definition of the CSP variables.

- Every ground state variable  $x_i$  whose value in  $s_0$  is  $v_i$  is encoded into a unary constraint of the corresponding CSP variable for  $j = 0$  of the form:

$$\{(x_i(0) = v_i)\} \quad (8.1)$$

- Every ground state variable  $x_i$  whose value is  $v_i$  in the goal  $g$  is encoded into a unary constraint of the corresponding CSP variable for  $j = k$

$$\{(x_i(k) = v_i)\} \quad (8.2)$$

Note that there is no constraint for  $s_0$  and  $g$  on the CSP variables  $\text{act}(j)$ .

**Example 8.15** The state  $s_0$  of Example 8.14 is translated into the following unary constraints:  $\{(\text{rloc}(0)=1)\}$ ,  $\{(\text{rload}(0)=\text{nil})\}$ , and  $\{(\text{cpos}(0)=12)\}$ .

The goal  $g$  is translated into the unary constraint:  $\{(\text{cpos}(4)=1)\}$ .  $\square$

**Step 3: Constraints encoding actions.** This encoding step translates the actions of the planning problem  $P$  into binary constraints of  $P'$ . We'll first define the set of all allowed pairs in these binary constraints for all actions; let  $E$  be this set. We'll then use the pairs in the set  $E$  to define the constraints on the state variables that encode each action.

For every action  $a(v_u, \dots, v_w)$ , an instance of some operator  $o \in O$  such that the constants  $v_u, \dots, v_w$  meet the rigid relations in the preconditions of  $a$ , and for  $0 \leq j \leq k - 1$ , we do the following:

- For every condition of the form  $(x_i = v_i)$  in  $\text{precond}(a)$  we put in  $E$  the pair:

$$(\text{act}(j) = a(v_u, \dots, v_w), x_i(j) = v_i) \quad (8.3)$$

- For every condition of the form  $(x_i = v_i)$  in  $\text{precond}(a)$  such that there is no assignment of  $x_i$  in  $\text{effects}(a)$  we put in  $E$  the pair:

$$(\text{act}(j) = a(v_u, \dots, v_w), x_i(j + 1) = v_i) \quad (8.4)$$

- For every assignment  $x_i \leftarrow v_i$  in  $\text{effects}(a)$  we put in  $E$  the pair:

$$(\text{act}(j) = a(v_u, \dots, v_w), x_i(j+1) = v_i) \quad (8.5)$$

Once this is done for all actions, we have in  $E$  all the pairs of allowed values for the variable  $\text{act}(j)$  and for the variables appearing in actions. The binary constraint on  $\text{act}(j)$  and a variable  $x$  is simply the union of all pairs in  $E$  related to  $\text{act}(j)$  and  $x$ .

**Example 8.16** The `move` and `load` operators in the Example 8.14 lead to the following pairs in  $E$ :

$$\begin{aligned} &(\text{act}(j) = \text{move}(l, m), \text{rloc}(j) = l), (\text{act}(j) = \text{move}(l, m), \text{rloc}(j+1) = m), \\ &(\text{act}(j) = \text{load}(l), \text{rloc}(j) = l), (\text{act}(j) = \text{load}(l), \text{rloc}(j+1) = l) \\ &(\text{act}(j) = \text{load}(l), \text{cpos}(j) = l), (\text{act}(j) = \text{load}(l), \text{cpos}(j+1) = \text{r1}), \\ &(\text{act}(j) = \text{load}(l), \text{rload}(j) = \text{nil}), (\text{act}(j) = \text{load}(l), \text{rload}(j+1) = \text{c1}), \end{aligned}$$

for  $(l, m)$  being either  $(l1, l2)$  or  $(l2, l1)$ , and for  $0 \leq j \leq 3$ . A similar set of pairs is defined for the operator `unload`.

The constraint between the two state variables  $\text{act}(j)$  and  $\text{rloc}(j)$ , for  $0 \leq j \leq 3$ , is the union of all the pairs in  $E$  related to these two variables, that is:

$$\begin{aligned} &\{(\text{act}(j) = \text{move}(l, m), \text{rloc}(j) = l), (\text{act}(j) = \text{load}(l), \text{rloc}(j) = l), \\ &(\text{act}(j) = \text{unload}(l), \text{rloc}(j) = l) \mid (l, m) \in \{(l1, l2), (l2, l1)\}\} \quad \square \end{aligned}$$

**Step 4: Constraints encoding frame axioms.** A variable that is invariant for an action  $a$  remains unchanged between  $s$  and  $\gamma(s, a)$ . Frame axioms can be encoded directly into a *ternary* constraints whose tuples involve  $\text{act}(j)$ , an invariant variable  $x_i(j)$  and  $x_i(j+1)$ . As in the previous step, we'll first define the set of all such tuples of possible values: for every action  $a(v_u, \dots, v_w)$  and every variable  $x_i$ , invariant for  $a$ , a tuple of possible values is:

$$(\text{act}(j) = a(v_u, \dots, v_w), x_i(j) = v_i, x_i(j+1) = v_i), \text{ for } v_i \in D_i \quad (8.6)$$

A frame axiom constraint is the union of all such tuples related to the same three variables  $\text{act}(j)$ ,  $x_i(j)$  and  $x_i(j+1)$ .

Note that `noop` has no action constraint, since it has no precondition and no effect, but every state variable is invariant for `noop`.

**Example 8.17** In Example 8.14, the variable `rload` is invariant with respect to the actions `move` and `noop`. Consequently, the frame axiom constraint for  $\text{act}(j)$ ,  $\text{rload}(j)$  and  $\text{rload}(j+1)$ , for  $0 \leq j \leq 3$ , is the following:

$$\begin{aligned} &\{(\text{act}(j) = \text{move}(l, m), \text{rload}(j) = v, \text{rload}(j+1) = v), \\ &(\text{act}(j) = \text{noop}, \text{rload}(j) = v, \text{rload}(j+1) = v) \mid v \in \{\text{c1}, \text{nil}\}\}. \quad \square \end{aligned}$$

**Plan extraction.** We have encoded a planning problem  $P$  and an integer  $k$  into a CSP  $P'$ . Let us assume that we have a tool for solving CSPs. Given  $P'$  as input, this CSP solver returns a tuple  $\sigma$  as a solution of  $P'$ , or “failure” if  $P'$  has no solution. The tuple  $\sigma$  gives a value to every CSP variable in  $P'$ , in particular to the variables  $\text{act}(j)$ . Let these values in  $\sigma$  be:  $\text{act}(j) = a_{j+1}$ , for  $0 \leq j \leq k-1$ . Each  $a_j$  is an action of  $P$ , and the sequence  $\pi = \langle a_1, \dots, a_k \rangle$  is a valid plan of  $P$  that possibly includes *noop* actions.

**Proposition 8.18** *There is a one-to-one mapping between the set of plans of length  $\leq k$  that are solutions of a bounded planning problem  $P$  and the set of solutions of the CSP problem  $P'$  encoding  $P$ .*

**Proof** Let  $\sigma$  be a tuple solution of  $P'$ . The value in  $\sigma$  of the variable  $\text{act}(0)=a_1$  meets all the constraints, in particular those specified through Equation 8.3: for every condition  $x_i(0) = v_i$  in  $\text{precond}(a_1)$ , the constraint between  $\text{act}(0)$  and  $x_i(0)$  allows only the value  $x_i(0) = v_i$  whenever  $\text{act}(0)=a_1$ . These values of the variables  $x_i(0)$  also meet the unary constraints in Equation 8.1 for state  $s_0$ . Consequently action  $a_1$ , whose preconditions are met in state  $s_0$  is applicable to  $s_0$ .

Consider now the state  $s_1$  corresponding to the state variables defined by the values of  $x_i(1)$  in the solution  $\sigma$ . These values of  $x_i(1)$  meet all the constraints, in particular those specified through Equations 8.4, 8.5, and 8.6 : whenever  $\text{act}(0)=a_1$  the only allowed values for  $x_i(1)$  are either those specified in the effects of  $a_1$ , or  $x_i(1) = x_i(0)$  when  $x_i$  is invariant in  $a_1$ . This is exactly the definition of the state resulting from applying  $a_1$  to  $s_0$ , hence  $s_1 = \gamma(s_0, a_1)$ .

The same argument applies for  $\text{act}(1)= a_2$  and  $s_2 = \gamma(s_1, a_2)$ ; it can be repeated till  $\text{act}(k-1) = a_k$  and  $s_k = \gamma(s_{k-1}, a_k)$ . Now, the values  $x_i(k) = v_i$  meet also the unary constraints of Equation 8.2, i.e., the goal  $g$  is satisfied in  $s_k$ . Hence  $\pi = \langle a_1, \dots, a_k \rangle$  is a valid plan of  $P$ .

Conversely, let  $\pi = \langle a_1, \dots, a_k \rangle$  be a solution plan of  $P$  and let  $s_1 = \gamma(s_0, a_1)$ ,  $\dots$ ,  $s_k = \gamma(s_{k-1}, a_k)$  be the corresponding sequence of states. Consider the tuple  $\sigma$  that gives to every CSP variable  $x_i(j)$  the value corresponding to that of the state variable  $x_i$  in state  $s_j$ , for  $0 \leq j \leq k$ , and  $\text{act}(j) = a_{j+1}$ , for  $0 \leq j \leq k-1$ . It is straightforward to show that  $\sigma$  meets all the constraints of  $P'$ , hence it is a solution of  $P'$ .

This proof also shows that there is no plan of length  $\leq k$  for the planning problem  $P$  iff the CSP  $P'$  is inconsistent.  $\square$