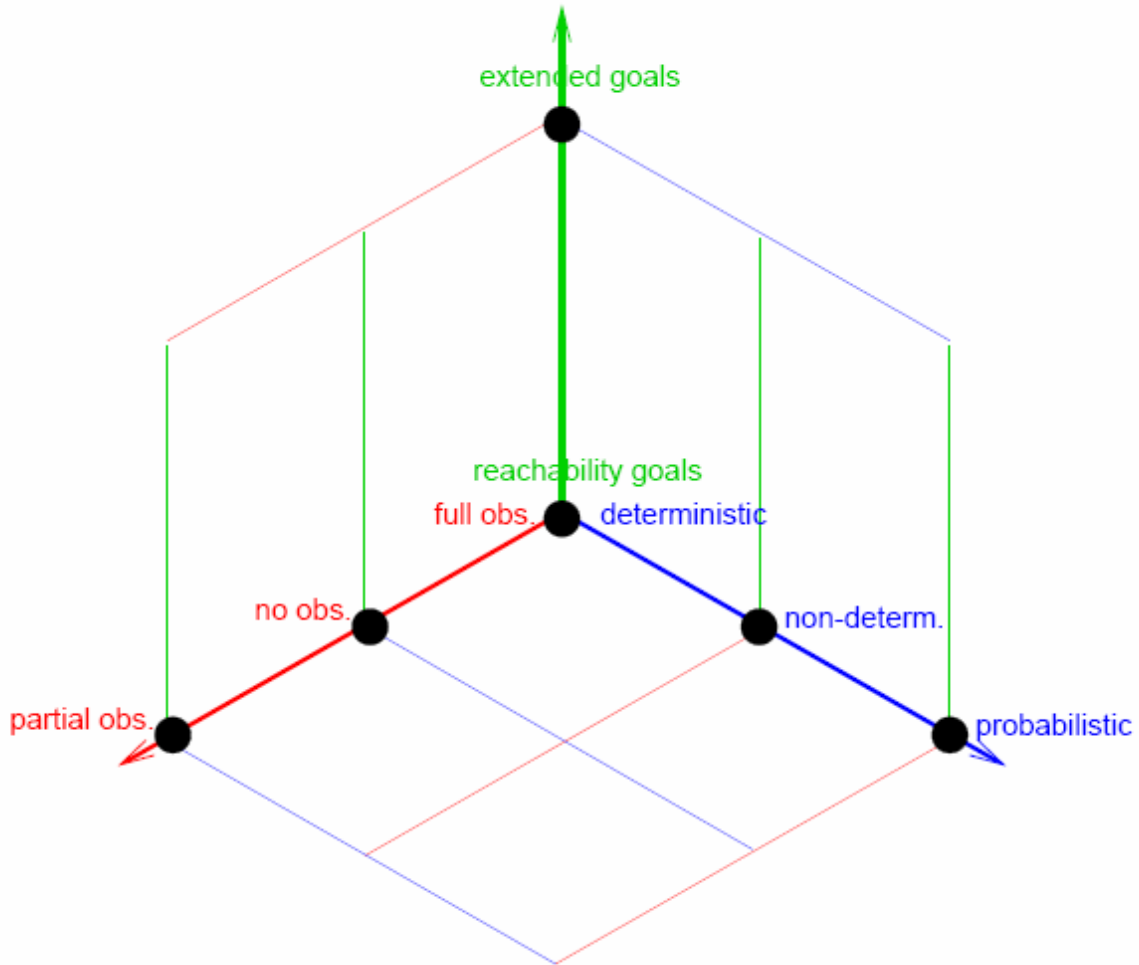


# “Planning based on Model Checking”

Summary of Chapter 17 and Appendix C  
 in Malik GHALLAB, Dana NAU and Paolo TRAVERSO: Automated Planning - Theory and Practice<sup>1</sup>

by Jan Priegnitz, Applied Systems Science  
 in the seminar “Planning Systems”, Prof. Hertzberg



outline<sup>2</sup>

<i>Introduction: Dimensions of Uncertainty</i> .....	2
<i>Nondeterministic Systems</i> .....	2
<i>Extended Goals</i> .....	7
<i>Partial Observability</i> .....	11

<sup>1</sup> I took some inspiration from the slides of Dana NAU, too. They are under a Creative Commons License.  
<http://www.laas.fr/planning/nau-lecture-slides.html>

<sup>2</sup>The footnotes will contain remarks about possible editorial improvements, but may be due to misunderstandings, too. Besides they serve as a hiding-place for the solutions of the exercises. Trivial orthography mistakes are listed at the end of the paper (a playful hunting to motivate me for thorough reading).

## Introduction: Dimensions of Uncertainty

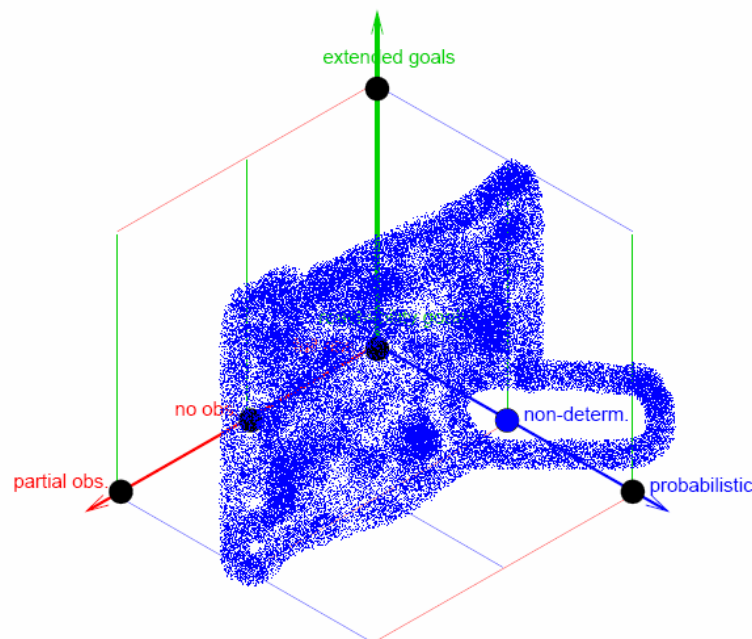
I want to start with the context of the topic: In this chapter the concept of uncertainty is introduced for the first time. Part V of the book begins with a figurative diagram of the possible dimensions of uncertainty<sup>3</sup> as shown on the cover of this paper.

Going out of the back corner<sup>4</sup> of the cube relaxes several assumptions which are valid for the restricted model presented in chapter 1 (pp. 11-15). These assumptions are A2, A4 and A1 about determinism, goals and observability respectively:

- The deterministic<sup>5</sup> model in A2 involved that only one action (if at all) is applicable to a state and that the outcome is definite.
- Restricted goals (A4) say that only the achievement of a goal state is necessary: Don't be diverted by the circumstances of your journey.
- In the restricted model assumption A1 allowed complete knowledge. That means the observation function is identical with the identity function (no noise in perception).

The relaxation of these assumptions induces the relaxation of assumption A5, too. A5 stated straightforward sequential plans for the restricted model. Now, with the use of conditional plans the path may branch.<sup>6</sup>

The previous chapter (not treated in the seminar) dealt with Markov Decision Processes (MDPs) which can be located on the right front side of the cube. Just as this probabilistic MDP level, planning systems with no observability are not a subject in this chapter (one may ask for the benefit of a blind robot locked in a black box).



## Nondeterministic Systems

In nondeterministic systems the planning agent remains uncertain, because more than one path of development is possible. In contrast a plan within classical planning is already its actual execution<sup>7</sup>.

<sup>3</sup> I would prefer a different order on the axis: On the axis concerning observability the order full, partial and no observability and on the determinism-axis the order deterministic, probabilistic and non-deterministic. That would be more intuitive to me. Otherwise the presented possible "space" of opportunities for designing a planning system is discrete rather than continuous, so a monotonous order was perhaps not borne in mind.

<sup>4</sup> It's all the same, whether you see this corner in the foreground or background ☺.

<sup>5</sup> There was made a distinction between the German terms 'deterministisch' and 'determiniert' in the lecture Informatik B: The latter one still takes into account the initial state, whereas 'deterministisch' even means the independence from the past. Assuming the same difference in the English language 'determined' would be more appropriate.

<sup>6</sup> On page 15 nondeterministic systems are linked with assumption A3. Obviously a mistake (A3 is listed again). A2 is proper.

Thus the question if a goal is reachable does not only mean to determine whether there is a topological solution in the maze of state-transitions, but to take into account that an action may fail regarding the intended outcome. For example a gripper might drop its load, or exogenous events like the closing of a road might disturb the domain. Furthermore EMERSON [173] argues that nondeterminism is essential for concurrent computation so that tasks share calculator resources equally (so-called fair scheduling assumption).

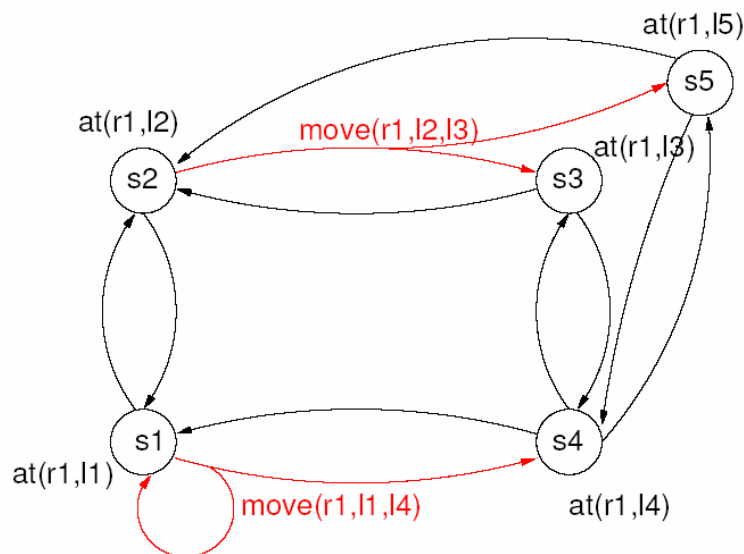
The definition of a planning domain<sup>8</sup>: a nondeterministic state-transition system  $\Sigma = (S, A, \gamma)$  where:

- S is a finite set of states
- A is a finite set of actions
- $\gamma: S \times A \rightarrow 2^S$  is the state-transition function (returns a power set of states)

The states are distinguished by the propositions holding in them. An action a is applicable in a state s if  $\gamma(s,a)$  is not empty. The possible actions for a state s are those for which a successor state exists:

$$A(s) = \{a: \exists s' \in \gamma(s, a)\}.$$

A planning problem P is a triple  $(\Sigma, S_0, S_g)$  with the sets of initial and goal states (compare with classical planning on p.19).



In the examples for nondeterministic systems and systems with extended goals the DWR domain (dock worker robots, but actually it's only one) is used. It was already introduced in the chapter about MDPs, but here no probabilities are assigned to the state-transitions. The locations  $l_x$  correspond to the states  $s_x$ . The actions are moves<sup>9</sup> between these locations. In the figure the arcs follow the execution of an action and branch if they are nondeterministic (see the red arcs). A 'policy'  $\pi$  for a planning domain  $\Sigma$  is less firm than a 'plan'<sup>10</sup>. The set of states of a policy is  $S_\pi = \{s \mid (s, a) \in \pi\}$ <sup>11</sup>. So we need an observing control in the execution of the policy (compare p.422 for MDPs).

```

Execute-Policy( $\pi$ )
  observe the current state s
  while s  $\in$  S $_\pi$  do
    select an action a such that (s, a)  $\in$   $\pi$ 
    execute action a
    observe the current state s
  end

```

<sup>7</sup> Think of the demon of LAPLACE! You can't even avoid thinking about him ☹. (And I'm not really wondering about my sentences – being a tiny machine in the clockwork of the universe I have no influence on the appearance of this seminar paper.)

<sup>8</sup> The Table of Notation on p. xix and xx may be worth a look.

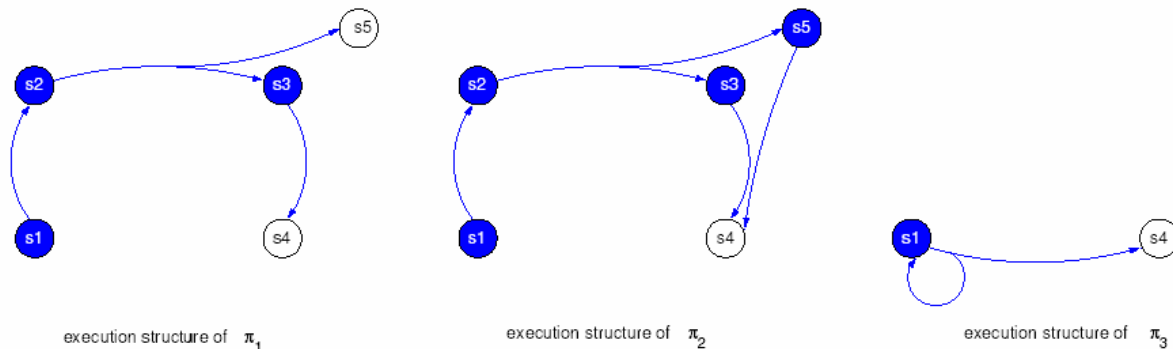
<sup>9</sup> The declaration of a  $move(r1, l_x, l_y)$  is a bit verbose, because it's always robot No.1.

<sup>10</sup> Nevertheless they use the terms policy and plan interchangeably.

<sup>11</sup> The sentence "We require that for any state s there is at most one action a such that (s, a)  $\in$   $\pi$ ." is misleading: The policy only contains the states in  $S_\pi$ , not 'any' state.

The following policies are different strategies for going from the initial state  $s_1$  to the goal state  $s_4$ .

- $\pi_1 = \{(s_1, \text{move}(r_1,11,12)), (s_2, \text{move}(r_1,12,13)), (s_3, \text{move}(r_1,13,14))\}$
- $\pi_2 = \{(s_1, \text{move}(r_1,11,12)), (s_2, \text{move}(r_1,12,13)), (s_3, \text{move}(r_1,13,14)), (s_5, \text{move}(r_1,15,14))\}$
- $\pi_3 = \{(s_1, \text{move}(r_1,11,14))\}$



In the figure they are depicted as ‘directed subgraphs’ of the whole graph (the terminal states are white). The reachability of a state can be easily checked.

When the robot performs  $(s_2, \text{move}(r_1,12,13))$  it can go astray to 15.  $\pi_1$  does not tell the robot what to do here. It is enlarged in  $\pi_2$  by  $(s_5, \text{move}(r_1,13,14))$  in order to cope with the branching path.  $\pi_3$  tries the short cut and can get into an infinite execution path which keeps the robot in the starting position.

The different policies correspond to different kinds of solutions:

- $\pi_1$  is a weak solution: At least one finite execution path reaches a goal.
- $\pi_2$  is a strong solution: Every execution path reaches a goal and is finite.
- $\pi_3$  is a strong cyclic solution: every ‘fair’ execution path reaches a goal. The ‘fairness’ assumption means that you don’t stay in a loop forever, if there’s a state-transition out of it. It is so to speak an iterative trial-and-error.

The set of strong solutions to a planning problem is a subset of the set of strong cyclic solutions, which in turn is a subset of the set of weak solutions:  $\Pi_{\text{strong}} \subseteq \Pi_{\text{strong-cyclic}} \subseteq \Pi_{\text{weak}}$

At this point I want to fit in an exercise (solution hidden in the footnote<sup>12</sup>):

„17.1 Consider Example 17.1 [above] with the goal to move the robot to state  $s_4$ . Let the set of initial states be  $s_1$  and [!!!]  $s_2$ . Are  $\pi_1, \pi_2, \pi_3$  weak, strong, strong cyclic solutions?“

In the next part algorithms are presented that are able to generate weak, strong and strong cyclic solutions. We start with strong planning.

```

Strong-Plan( $P$ )
   $\pi \leftarrow \text{failure}; \pi' \leftarrow \emptyset$ 
  While  $\pi' \neq \pi$  and  $S_0 \not\subseteq (S_g \cup S_{\pi'})$  do
     $\text{PreImage} \leftarrow \text{StrongPreImg}(S_g \cup S_{\pi'})$ 
     $\pi'' \leftarrow \text{PruneStates}(\text{PreImage}, S_g \cup S_{\pi'})$ 
     $\pi \leftarrow \pi'$ 
     $\pi' \leftarrow \pi' \cup \pi''$ 
  if  $S_0 \subseteq (S_g \cup S_{\pi'})$  then return( $\text{MkDet}(\pi')$ )
  else return( $\text{failure}$ )
end
  
```

There are three variables  $\pi, \pi'$  and  $\pi''$ ;  $\pi$  is initialized with failure,  $\pi'$  is empty. The while-loop stops if there aren’t any changes any more (failure) or all initial states are included in the union of sets of the goal states and the states in the found policy (success). It is a backward breadth-first search starting with the goal states until all initial states appeared among the predecessors. The first method

$$\text{StrongPreImg}(S) = \{(s, a): \gamma(s, a) \neq \emptyset, \gamma(s, a) \subseteq S\}$$

<sup>12</sup>  $\pi_1$  and  $\pi_2$  remain weak and strong respectively, but  $\pi_3$  is not even weak, because this policy doesn’t take  $s_2$  into consideration.

returns only those preceding state-action pairs which finish in any case in the given set S. The second method

$$\text{PruneStates}(\pi, S) = \{(s, a) \in \pi : s \notin S\}$$

removes those state-action pairs which start from the already solved set. The third method  $\text{MkDet}(\pi')$  converts a possibly nondeterministic policy into a deterministic one by removing superfluous state-action pairs (imagine a graph where two actions lead to different locations, but merge into one location later on).  $S_\pi$  is iteratively used as a target for  $\text{StrongPreImg}$  which returns new “slides”.

Now we will follow this algorithm step by step for the problem in the DWR domain. After the instantiation the while-statement is passed ( $\emptyset \neq \text{failure}$  and  $s1 \not\subset s4$ ).  $\text{StrongPreImg}$  finds  $\{(s3, \text{move}(r1, l3, l4)), (s5, \text{move}(r1, l5, l4))\}$  and suppresses  $(s1, \text{move}(r1, l1, l4))$ , because l1 does not belong to  $S_g$ .  $\text{PruneStates}$  has nothing to do, because s3 and s5 are not part of the goal set ( $S_{\pi'}$  is still empty). The  $\pi$  is updated with the empty policy,  $\pi'$  gets the two state-action pairs from  $\pi'$ . So they differ for the second iteration and  $S_g \cup S_{\pi'} = \{s3, s4, s5\}$  lacks for s1.

Again  $\text{StrongPreImg}$  delivers the two state-action pairs and in addition  $\{(s2, \text{move}(r1, l2, l3)), (s4, \text{move}(r1, l4, l3)), (s4, \text{move}(r1, l4, l5))\}$ . The last two do not make progress and are pruned immediately. The first two, too, since  $S_{\pi'}$  contains their starting states. It would be nonsensical to add them twice to  $\pi'$  which is now  $\pi' = \{(s2, \text{move}(r1, l2, l3)), (s3, \text{move}(r1, l3, l4)), (s5, \text{move}(r1, l5, l4))\}$ .

In  $S_g \cup S_{\pi'} = \{s2, s3, s4, s5\}$  only s1 is lacking, but will appear in  $\text{PreImage}$ .

After pruning,  $(s1, \text{move}(r1, l1, l2))$  is the only new state-action pair. So  $\pi'$  equals the former  $\pi_2$  ( $\text{MkDet}(\pi')$  is satisfied).

Weak planning differs only in one detail:

$$\text{WeakPreImg}(S) = \{(s, a) : \gamma(s, a) \cap S \neq \emptyset\},$$

so there has to be at least one successor in S.

Given the DWR problem there is a little surprise:  $\text{WeakPreImg}$  returns  $\{(s1, \text{move}(r1, l1, l4)), (s3, \text{move}(r1, l3, l4)), (s5, \text{move}(r1, l5, l4))\}$ , nothing is pruned and s1 is already the initial state. Two moves are useless, but don't disturb either. So it is the strong cyclic  $\pi_3$  which is no error, because  $\Pi_{\text{strong-cyclic}} \subseteq \Pi_{\text{weak}}$ . Weak planning always returns the shortest policy.

Paper [129]<sup>13</sup> proves that the algorithms are sound (correctness, termination, optimality). The authors define the term ‘trace’ as a sequence of states connected by transitions (I preferred ‘path’). The weak distance of a state s is the minimal number of transitions in a trace to go from s to an state  $\in S_g$  (when  $s \in S_g$  the weak distance is 0, when there is no trace at all:  $\infty$ ). One can state that the states with the weak distance i are added in the i-th iteration. For correctness and termination they argue with the finitely many state-action pairs, for optimality they use the above statement about the iterative enlargement. (The proofs for strong planning need another term: a ‘complete set of traces’ covers all possible outcomes of actions, its length is that of the longest trace. The strong distance is now the longest trace in the shortest complete set of traces (when there is no such set at all:  $\infty$ ). Again state-action pairs with the strong distance i are added in the i-th iteration.)

When we apply these two algorithms in deterministic classical planning, it turns out that they behave in the same manner ( $\Pi_{\text{strong}} = \Pi_{\text{weak}}$ ).

```

Strong-Cyclic-Plan( $S_0, S_g$ )
   $\pi \leftarrow \emptyset$ ;  $\pi' \leftarrow \text{UnivPol}$ 
  while  $\pi' \neq \pi$  do
     $\pi \leftarrow \pi'$ 
     $\pi' \leftarrow \text{PruneUnconnected}(\text{PruneOutGoing}(\pi', S_g), S_g)$ 
  if  $S_0 \subseteq (S_g \cup S_{\pi'})$ 
    then return( $\text{MkDet}(\text{RemoveNonProgress}(\pi', S_g))$ )
    else return(failure)
end

```

<sup>13</sup> Although it provides the same theory it has no allusion to robots. The authors are very busy cooking omelettes...

```

PruneOutgoing( $\pi, S_g$ ) ;; removes outgoing state-action pairs
 $\pi' \leftarrow \pi - \text{ComputeOutgoing}(\pi, S_g \cup S_\pi)$ 
return( $\pi'$ )
end

PruneUnconnected( $\pi, S_g$ ) ;; removes unconnected state-action pairs
 $\pi' \leftarrow \emptyset$ 
repeat
 $\pi'' \leftarrow \pi'$ 
 $\pi' \leftarrow \pi \cap \text{WeakPreImg}(S_g \cup S_{\pi'})$ 
until  $\pi'' = \pi'$ 
return( $\pi'$ ) end

RemoveNonProgress( $\pi, S_g$ ) ;; remove state-action pairs that
;; do not lead towards the goal

 $\pi^* \leftarrow \emptyset$ 
repeat
 $\text{PreImage} \leftarrow \pi \cap \text{WeakPreImg}(S_g \cup S_{\pi^*})$ 
 $\pi_{old}^* \leftarrow \pi^*$ 
 $\pi^* \leftarrow \pi^* \cup \text{PruneStates}(\text{PreImage}, S_g \cup S_{\pi^*})$ 
until  $\pi_{old}^* = \pi^*$ 
Return( $\pi^*$ )
end

```

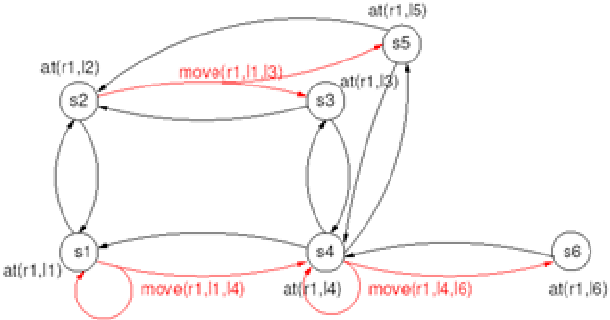
The algorithm for strong cyclic planning has a different approach, but falls back on the methods already explained. The all-embracing ‘universal policy’ =  $\{(s,a) \mid a \in A(s)\}$  is decreased in several steps. The while-loop is waiting for a fixed point. Meanwhile an encapsulated method

$$\text{ComputeOutgoing}(\pi, S_g \cup S_\pi) = \{(s, a) \in \pi \mid \gamma(s, a) \not\subset S_g \cup S_\pi\}^{14}$$

looks for state-action pairs that are impasses and removes them inside PruneOutgoing. The result is passed to PruneUnconnected. During this shrinking of the ‘universal policy’ fragments without a goal state can emerge. By intersecting the current policy with the known WeakPreImg we play safe that there is no chance to escape from these hopeless islands – they are removed, too. (But it is possible to travel within them, otherwise they would have been already removed by PruneOutgoing. Connections between states into both directions ‘take care of each other’.)

RemoveNonProgress avoids backward steps and involves PruneStates.

The example is extended by a new goal state s6 (see figure). The while-loop is left immediately, because the graph is completely joined bidirectional. But RemoveNonProgress complains about  $(s4, \text{move}(r1,14,11))$ ,  $(s4, \text{move}(r1,14,13))$ ,  $(s4, \text{move}(r1,14,15))$ ,  $(s3, \text{move}(r1,13,12))$ ,  $s5, \text{move}(r1,15,12))$ , and  $(s2, \text{move}(r1,12,11))$ , all turning their back on s6. For example it would be wrong to accept the move from 14 to 11. Strong cyclic does not mean that the actions are chosen randomly but that their outcome has not an unfair random distribution (the term random is a bit too reminiscent of MDPs). On the other side this move would be acceptable if it branches to the goal state. In other words: The robot has no ‘meta-policy’ for the actual policy when there is a choice of state-action pairs.



MkDet makes it deterministic and can pick one of the following orders: (11, 14, 16), (11, 12, 13, 14, 16) or (11, 12, 15, 14, 16).

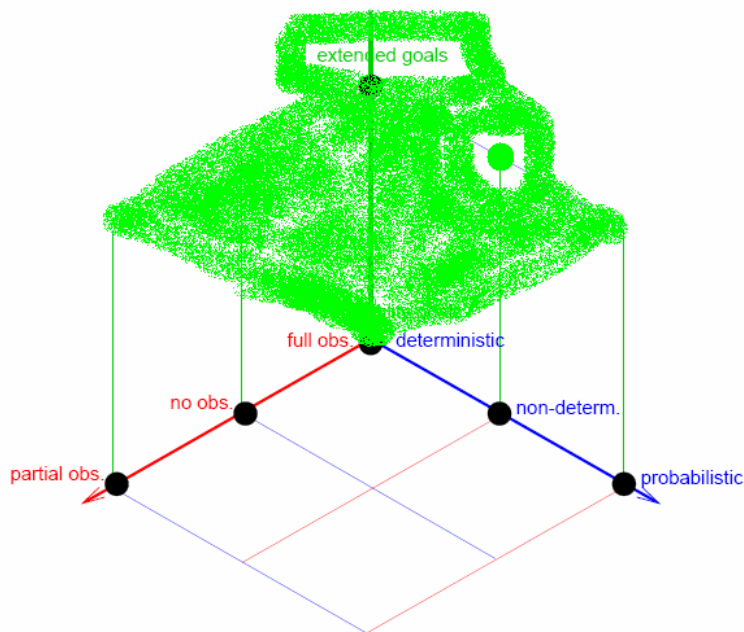
<sup>14</sup> First I got suspicious, because  $S_\pi$  is not passed to the surrounding method.

[129] presents a second algorithm, which they call ‘local’ instead of ‘global’. It has not the disadvantage of the global algorithm that does not take a strong solution if there is a shorter strong cyclic solution. Like in weak and strong planning a search backwards is done. A cyclic behaviour in the policy is only considered when a fixed point is reached in the strong extension. As soon as a non-empty strong cyclic extension is returned, the cyclic extension phase ends. Since the algorithm can only increase the quality of the returned policies they categorize it as a particular case of a family of "optimizing" algorithms.

In order to conclude this chapter, two more exercises (solutions hidden here<sup>15</sup>):

“17.2 In Example 17.1, with the goal to move the robot to state  $s_4$ , modify the action  $\text{move}(r1, l1, l4)$  such that its outcome can also be in state  $s_3$ . Write a weak, a strong and a strong cyclic solution (if they exist).”

“17.3 In Example 17.1, with the goal to move the robot to state  $s_4$ , modify the action  $\text{move}(r1, l1, l4)$  such that its outcome can also be in a dead end state  $s_6$ . Write a weak, a strong and a strong cyclic solution (if they exist).”



## Extended Goals

In this chapter the robot will not declare ‘mission accomplished’ – it is urged to have special tasks. If you want to command a robot to keep visiting several locations for ever and ever, a goal state set can not be sufficient. A first attempt could be changing this goal state set temporally, but since there is no moderator the robot has to pay attention itself. The presented solution is to give it a small memory called context. Hence it knows what having done and what to do next. After a subgoal has been solved it turns into the new context.

Furthermore we still have to deal with nondeterminism (how it is indicated in the graph, too). So we will have to content ourselves that the robot does only ‘its best’ to follow our command. This seems to be a little vague at first, but will be formalized with Computation Tree Logic.

Here the term ‘policy’ is taken over from the term ‘plan’, with this definition: It is a tuple  $(C, c_0, \text{act}, \text{ctxt})$ , where

- $C$  is a set of execution contexts
- $c_0$  is the initial context

<sup>15</sup> for both exercises: weak =  $\{(s1, \text{move}(r1, l1, l4))\}$  (former  $\pi_3$ )  
and strong =  $\{(s1, \text{move}(r1, l1, l2)), (s2, \text{move}(r1, l2, l3)), (s3, \text{move}(r1, l3, l4)), (s5, \text{move}(r1, l5, l4))\}$  (the same  $\pi_2$ )  
for 17.2: strong cyclic =  $\{(s1, \text{move}(r1, l1, l4)), (s3, \text{move}(r1, l3, l4))\}$   
for 17.3: strong cyclic =  $\{(s1, \text{move}(r1, l1, l2)), (s2, \text{move}(r1, l2, l3)), (s3, \text{move}(r1, l3, l4)), (s5, \text{move}(r1, l5, l2))\}$

- act:  $S \times C \rightarrow A$
- ctxt:  $S \times C \times S \rightarrow C$

In the context function the second S stands for the end state – several actions might be applicable and these might be subject to nondeterminism, too. We require  $\gamma$  being total, that means for every state exists an action. In contrast the action and context function are partial, some state-context combinations do not occur.

We consider only plans that are executable and complete. A plan is executable if the end state  $s'$  of every occurring action-context-function combination is offered in the belonging state-transition function. It is complete if there is always some context for each action-end state combination.

Once more we take the DWR, without  $s_6$ . A difficult task would be “keep moving back and forth between locations  $l_2$  and  $l_4$ , and never pass through  $l_5$ ”. It is like choosing between Scylla and Charybdis: Moving via  $l_3$  bears the risk landing in  $l_5$ , moving via  $l_4$  might take very long. We can weaken the requirement: Move to  $l_4$  ‘just if possible’. A corresponding plan is in the table:

<i>state</i>	<i>context</i>	<i>action</i>	<i>next state</i>	<i>next context</i>
s1	c1	move(r1,l1,l2)	s2	c2
s1	c2	move(r1,l1,l4)	s1	c1
s1	c2	move(r1,l1,l4)	s4	c1
s2	c2	move(r1,l2,l1)	s1	c2
s4	c1	move(r1,l4,l1)	s1	c1

Notice that the robot leaves the context after failing.

In the discussion after the presentation the suggestion was made to convert the state-context-pairs into more states without explicit contexts – feasible, but the structure may be confusing.

In this ‘context’ fits exercise 17.5 (curious about the solution?<sup>16</sup>)

<i>state</i>	<i>context</i>	<i>action</i>	<i>next state</i>	<i>next context</i>
s1	c0	move(r1,l1,l4)	s1	c1
s1	c0	move(r1,l1,l4)	s4	c4
s4	c0	wait	s4	c0
s1	c1	move(r1,l1,l2)	s2	c2
s2	c2	move(r1,l2,l1)	s1	c2
s1	c2	move(r1,l1,l2)	s2	c2

Let the initial state be  $s_1$  and the the goal state  $s_4$ . Is  $\pi$  a weak, strong, strong cyclic solution?

And, what about the following plan?

<i>state</i>	<i>context</i>	<i>action</i>	<i>next state</i>	<i>next context</i>
s1	c0	move(r1,l1,l4)	s1	c1
s1	c0	move(r1,l1,l4)	s4	c4
s4	c0	wait	s4	c0
s1	c1	move(r1,l1,l2)	s2	c2
s2	c2	move(r1,l2,l1)	s1	c0

<sup>16</sup> first some comments on the lines:

- move fails, jumps to line 4
- move succeeds in the goal state
- a pseudo-action, „all is well that ends well“
- only an evasive action: upwards
- downwards
- and up again... an endless cycle with no opportunity to escape

Therefore this plan is weak.

The following plan lacks the last line; the fifth is changed in the next context. So it does only one round and we give him one more try, and even more (we are generous), therefore: strong cyclic.

Information about Computation Tree Logic (CTL) is provided by Appendix C and [173], it features a condensed grammar for expressing goals. A CTL formula consists of a path quantifier and a temporal operator. The path quantifiers ‘A’ and ‘E’ mean that an expression ‘always’ or ‘sometimes’ hold respectively. They can be remembered easily by mirroring them:  $\forall$  and  $\exists$ . Primitive temporal operators are (their symbol **bold**) **neXt** and **Until**. All the others like **Weak-until**, **Future** and **Globally** can be derived from them. Past tense operators are missing in this book and can be indicated with a ‘-’ in <sup>superscript</sup>. For including only the present time slot write ‘ $\geq$ ’ instead of ‘ $>$ ’ ([173] calls it "reflexive" future).

CTL formulas are inductively defined:

1. Every atomic proposition  $p \in P$  is a CTL formula.
2. If  $p$  and  $q$  are CTL formulas, then so are the following formulas:
  - (a)  $\neg p$ ,  $p \vee q$ ,
  - (b)  $AXp$ ,  $EXp$ ,
  - (c)  $A(pUq)$ , and  $E(pUq)$

The binding power of the operators has this order (beginning with the highest):  
temporal operators,  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$ ,  $\Leftrightarrow$ .

Some exemplary explanations:  $EXp$  means that there exists a next state in which  $p$  holds.  $A(pUq)$  means that in all paths  $p$  holds (as a prefix) until  $q$  appears. **Future** (or ‘eventually’) and **Globally** (or ‘always’) are abbreviations:  $AFp$  corresponds to  $A(\top U p)$ ,  $EFp$  to  $E(\top U p)$ ,  $AGp$  to  $\neg EF\neg p$  [=  $\neg E(\top U \neg p)$ ] and  $EGp$  to  $\neg AF\neg p$  [=  $\neg A(\top U \neg p)$ ]. ‘ $\top$ ’ is a true proposition. Other common expressions are  $pBq$  [=  $\neg(\neg p U q)$ ],  $p$  precedes  $q$ , or: ‘somewhere before’ and  $F^\infty p$  [=  $GFp$ , ‘infinitely open’] and  $G^\infty p$  [=  $FGp$ , ‘almost everywhere’].

The translations for a weak, strong or strong cyclic goal  $g$  in CTL are  $EFg$ ,  $AFg$  and  $A(EFg Wg)$ .

**Weak-until** (in comparison to the strong **Until** which does not allow infinite paths) is used like this:  $A(pWq)$  is satisfied if  $p$  holds forever or it holds until  $q$  holds<sup>17</sup>.  $U$  and  $W$  can be transferred mutually:  $pUq = (pWq \wedge Fq)$  and  $pWq = (pUq \vee Gp)$ .

Now we are able to express the ‘difficult task’ much shorter:

$$AG(AF \text{ at}(r1,l2) \wedge EF \text{ at}(r1,l4))$$

EMERSON [173] presents a classification of temporal logics. The distinguishing features are:

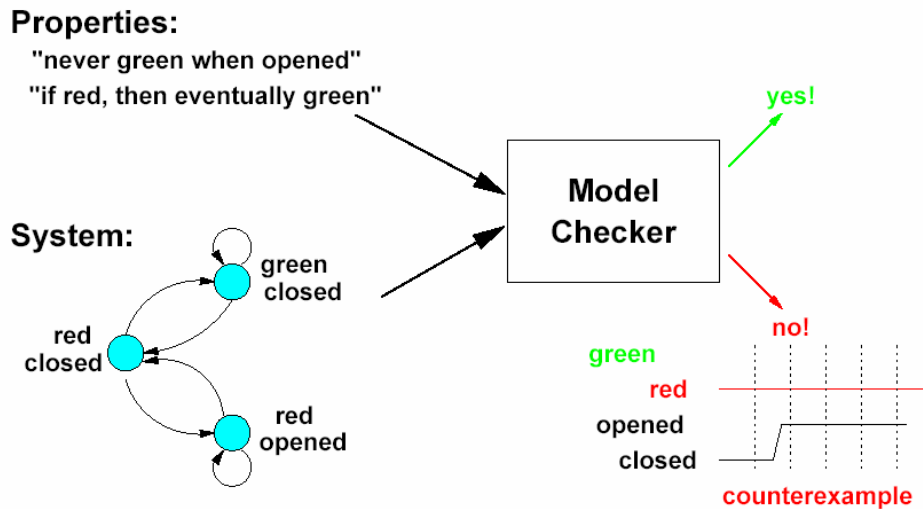
- propositional vs. first-order
- global vs. compositional
- branching vs. linear (the realm of nondeterminism)
- points vs. intervals
- discrete vs. continuous<sup>18</sup>
- past vs. future tense

The path quantifiers are crucial in CTL and make it more difficult than Linear Temporal Logic (LTL).

The validation of a plan is done by ‘model checking’. This term is used in various contexts abundantly. It checks if a finite structure is a model of the formula. This diagram represents a railroad crossing with three states. The red-green-property is important for the train driver, the open-close-property for the person on the road. The liveness-requirement (guarantees that the system does not get stuck) and the strong cyclic concept resemble one another.

<sup>17</sup> in the book: ‘...until  $p$  holds.’

<sup>18</sup> His reason for preferring a discrete timeline is the discrete way of working in computers, but this reason only holds when planning in this domain. In the distant future when the division of time by computers outperforms the PLANCK-time (=  $10^{-43}$  s) the need for a discrete real-time model may arise again ☺.



For the CTL-EFp the model checking algorithm looks like this:

1.  $MCHECKEF(p,K)$
2.  $CurrentStates \leftarrow \emptyset;$
3.  $NextStates \leftarrow STATES(p,K);$
4. while  $NextStates \neq CurrentStates$  do
5.   if  $(S_0 \subseteq NextStates)$
6.    then return(*True*);
7.    $CurrentStates \leftarrow NextStates;$
8.    $NextStates \leftarrow NextStates \cup PRE-IMG-EF(NextStates,K);$
9.   return(*False*);

$$PRE-IMG-EF(States,K) = \{s \in S : \exists s'. (s' \in States \wedge R(s,s'))\}$$

For MCHECKAF replace this method with

$$PRE-IMG-AF(States,K) = \{s \in S : \forall s'. (R(s,s') \rightarrow s' \in States)\}$$

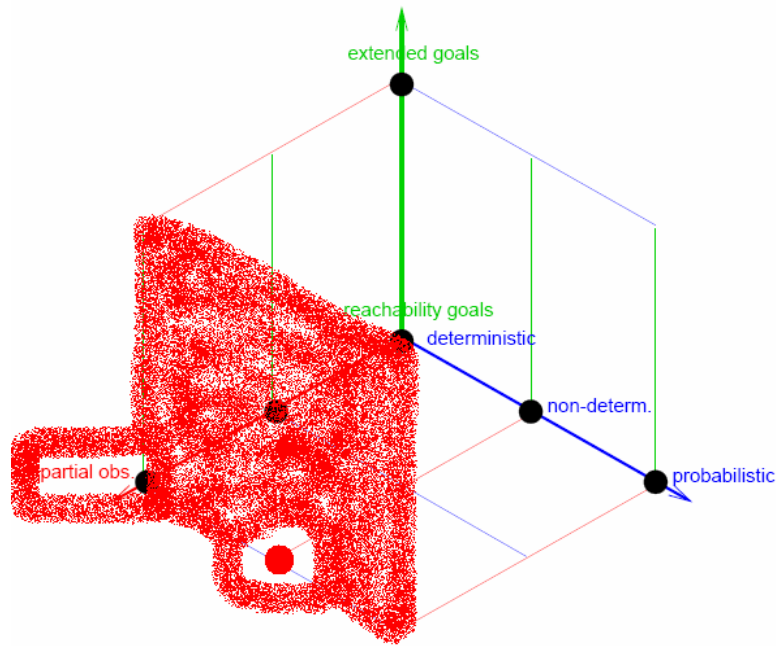
The similarity with the algorithms for weak and strong planning is obvious.

I want to skip the next domain example<sup>19</sup>.

The motivation for introducing the EAGLE language is that it is more powerful than CTL (and LTL, too) with regard to failure handling. The semantics enable expressions for preferences:

- reachability (basic) goals: DoReach p, TryReach p (similar to AFp and EFp, but considers failure and success)
- maintenance (basic) goals: DoMaintain p, TryMaintain p
- conjunction: g And g' (parallel)
- failure: g Fail g'
- control operators: g Then g', Repeat g (in a cyclic way)

<sup>19</sup> On the one hand it is very graphic, the naming is more vivid than I1, I2, I3... and would have served as a good start into the abstract execution structures. On the other hand it lacks just as much their clarity. It is not comprehensible at once that the robot is adjusted with a wrong driving angle and therefore drifts into the forbidden lab. And you have to look twice in order to realize the autonomous behaviour of the door. I suppose an error in this sentence: "If dep does not hold, then EFdep must be satisfied for some of the outcomes (the arc going back to the same [?other!] context), and for all the other outcomes that do not satisfy EF dep."



### Partial Observability

The robot is equipped with sensors. The first definition is only used in one example and then replaced by a second one. Here the first:

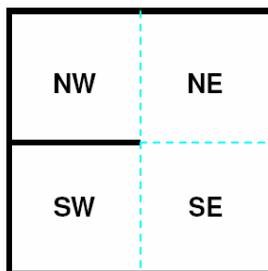
$\Omega$  is a finite set of observations. An observation function over  $\Omega$  and the state transition system  $S$  is a function  $O: S \rightarrow 2^\Omega$ , which associates to each state  $s$  the set of possible observations  $O(s) \subseteq \Omega$ . We require that for each  $s \in S$ ,  $O(s) \neq \emptyset$ .

The case of incomplete information ( $O(s_1) = O(s_2)$ , with  $s_1 \neq s_2$ ) lies between the extreme cases of null ( $\Omega = \{o\}$ ,  $O(s) = \{o\} \forall s \in S$ ) and full observability ( $\Omega = S$ ,  $O(s) = \{s\}$ ).

The second definition:

$V$  is a finite set of observation variables. The evaluation of an observation variable  $v \in V$  is the relation  $Xv: S \times \{\tau, \perp\}$ . In  $v_\tau$  and  $v_\perp$  we collect the states in which the observation is true or false respectively. If  $v$  is undefined in a state  $s$ , then  $s \in v_\tau \cap v_\perp$ , that means both observations hold.

At last the traditional planning domain  $\Sigma$  can be extended to a tuple  $(\Sigma, V, Xv)$ .



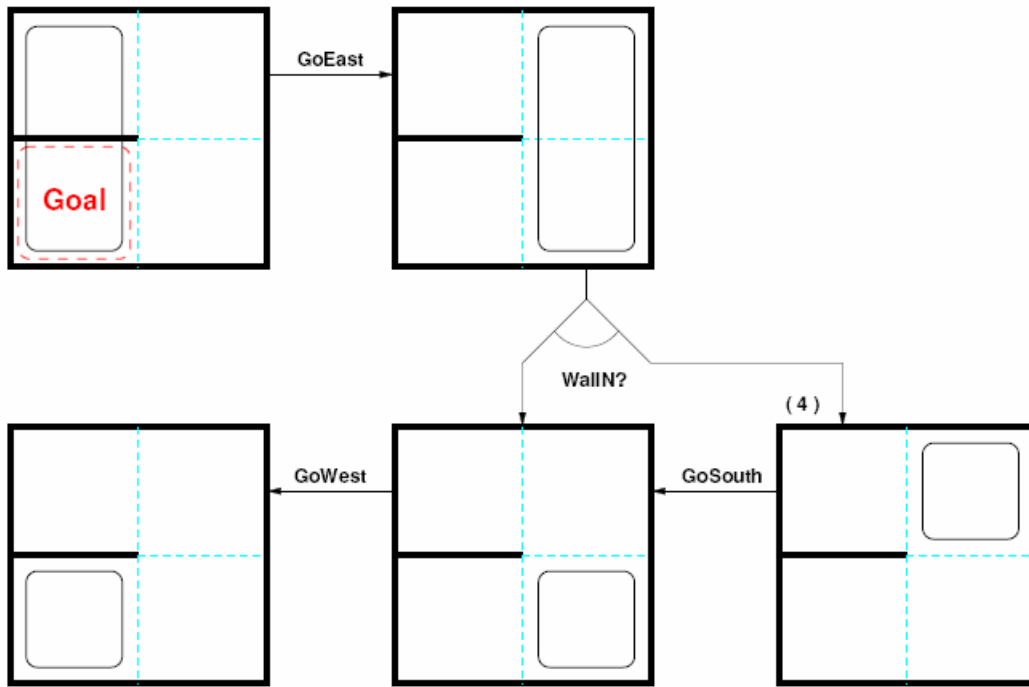
The example shows a very small maze. The states  $S$  are the corners  $\{NW, NE, SW, SE\}$ , the actions  $A$   $\{GoNorth, GoSouth, GoEst, GoWest\}$  are only possible if there is no wall. The observation variables  $V$  are  $\{WallN, WallS, WallW, WallE\}$ . Due to the limited range of the observations the states  $NW$  and  $SW$  look the same ( $X$  stands for  $N$  and  $S$ ):  $XWallN(YW, \tau)$ ,  $XWallW(YW, \tau)$ ,  $XWallS(YW, \tau)$ ,  $XWallE(YW, \perp)$ . For a longer observation range you can add the variables  $2WallY$  which are undefined if blocked by near walls.

Let  $S_0$  be  $NW$  or  $SW$ , a so-called belief state,  $S_g = \{SW\}$ . In  $S_0$  the robot does not know if it is already in  $S_g$ . An useful policy is (with  $\lambda$  as the empty policy):

GoEast; (if WallN then GoSouth else  $\lambda$ ); GoWest

In a belief state an action is only applicable iff it is applicable in all contained states.

The execution is shown here. After the observation the belief state implodes.

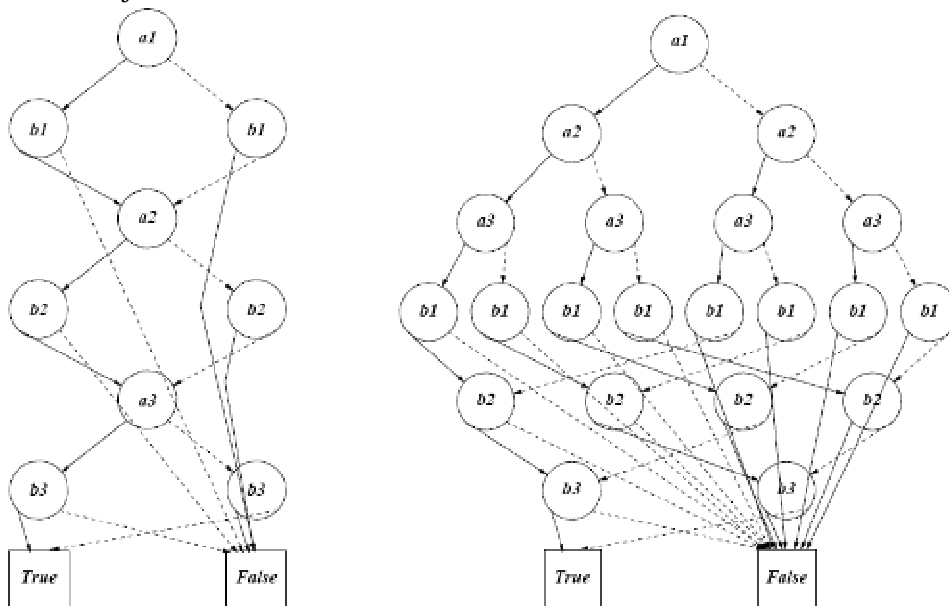


AND/OR graphs over belief states are suitable for obtaining policies.

The belief states already aggregated states. This approach is supported by **Symbolic Model Checking**, too. Instead of handling single states or transitions they are grouped to sets of states and sets of transitions respectively. State variables are stored in vectors.  $\xi(s)$  is the propositional representation of the characterizing truth values in  $s$ , or for any set of states  $Q \subseteq S$ :  $\xi(Q) = \bigvee_{s \in Q} \xi(s)$ .

This representation may be strikingly short for huge sets independent of the cardinality of the set of propositions  $P$ :  $\xi(2^P) = \top$ ,  $\xi(\emptyset) = \perp$ .

At last there are some remarks on **binary decision diagrams** (BDDs). A BDD is a directed acyclic graph (DAG). The terminal nodes are the truth values, each non-terminal node  $n$  is associated with a boolean variable  $\text{var}(n)$ . The ordering of variables may be important, e.g. both BDDs below stand for the formula  $(a1 \leftrightarrow b1) \wedge (a2 \leftrightarrow b2) \wedge (a3 \leftrightarrow b3)$ . It is possible to transform a BDD to obtain the BDD representing the negation of the formula or to combine two BDDs to obtain the BDD representing the conjunction or the disjunction of the two formulas.



It seems to be paradox that performances in planning by model checking become better with increasing uncertainty. MDPs are only practical, if probabilities are accessible, otherwise they would simulate an artificial accuracy. The authors maintain that planners based on symbolic model checking get better benchmarks than planners based on MDPs. So this paper presented more than a preliminary stage to MDPs. Paper [129] shows detailed tables for performance tests. The time needed for preprocessing was excluded, because the candidates used different techniques (not all had to generate and to compile the source code of the internal representations, reuse in new runs was not always possible – perhaps gradient and preprocessing-y-axis-section of a straight line equation can serve as indicators depending on the number of runs).

*additional literature*

[129] A. CIMATTI, M. PISTORE, M. ROVERI and P. TRAVERSO (2003): Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence*, 147(1-2):35–84, 2003.

[173] E. A. EMERSON (1990): Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, chapter 16, pages 995–1072. Elsevier, 1990.

*trivial orthography mistakes*

p. 13 “each ~~an~~ action can lead to...”

p. 452 “... $\pi$  ‘ may have more ~~that~~ [than] one action...”

p. 453 “are un[n]eeded”

p. 455 “PruneUnconnected removes ~~every~~ [all] edges that are ~~are~~ unconnected”

p. 457 “the basic assu[m]ption”; “exetension” is extended with an ‘e’; “of different stre[n]gth”

p. 458 “formalize ~~teh~~ [the] planning problem”

p. 459 “does not pass ~~thorough~~ [through]”; “a different ~~strenght~~ [strength]”; “we can si[y]nthesize”

p. 460 “ther[e] is a state”

p. 461 „ $AF(p \wedge AFq)$  can be uised to express the goal that p must become true and, after that p ~~becomes true, then~~ q becomes true.“; “we get out of ~~et~~ [it]”; “the (strong) until oper[a]tor”

p. 463 “and AFstore afterward[s]”; “and EFdep) afterward[s]”

p. 464 “if dep does not holds”; “...in the lab[.] At this step...”

p. 468 “and ~~and~~ an algorithm”

p. 469 “a finite set of ~~of~~ observations”

p. 470 “However, these observations can be easily modeled by representing explicitly ~~in~~ the state of the domain (the relevant information) on} the last executed action.” - a bit weird; “ $S = \{NW, NE, SW, SE, [^{20}]\}$ ”; “ $A = \{GoNorth, GoSouth, GoEst, GoWest, [^{21}]\}$ ”

p. 471 “In partially obbservable domains”

p. 472 “intuitive[l]y”

p. 477 “are incomparable with ~~the~~ those discussed in this chapter.”

p. 480 “and the ~~the~~ goal state”

Inversely I am waiting for your errata list. As a non-native English speaker I probably ‘exchanged’ these mistakes with much more.

---

<sup>20</sup> it is getting meticulously,...

<sup>21</sup> ... but I claim to have found them all