Planning as model checking

Omar El-khatib and Hung Viet Le

March 2, 2004

It is another way to generate plans (i.e. solve plan problems). The key idea is to represent planning problem as a [planning domain with properties of the domain][[*a model checking problem*]] then [verify that the properties are true in the planning domain][[*use one of the many model checking tools to solve it*]]. Model checking is the problem of determining if a *CTL* formula is true in the model or not. First we need to [define][[*spcify the language/model for the describing of*]] the domains and the properties of the domain. Domains are defined using Kripke Structures, and properties are defined by Computation Tree Logic (CTL) *formulas*.

1 Kripke Structure

A [domian in][[]] Kripke Structure K is given by a tuple of four components: <W, W_0, T, L>, some system adds one more component P which is the set of propositions, where:

- W: is a set of states.
- W_0: is a set of initial states which [are][[*is a*]] subset of W (i.e. $W_0 \subseteq W$).
- $T \subseteq W \times W$: is a [[*total*]] transition relation where [it should be total meaning that]:
  $\forall$ w $\in$ W, $\exists$ w' such that (w, w') $\in$ T
- L: is a labeling function, L:W $\rightarrow 2^P$ [were] [[*where*]] P is the [[*set of*]] atomic propositions. 2^P is the power set of the atomic proposition P. [i.e. L is a map from a set of states to set of all subsets of P, or L=$\{$s : s $\in$ P $\}$.] In otherwords, L assigns a set of atomic propositions (which is $\subseteq$ P) to each state.

A path is an infinite sequence of states $w_0 w_1 w_2 \ldots w_n \ldots$ where the first state $w_0 \in W_0$ (i.e. an initial states), and each $(w_i, w_{i+1}) \in$ T. Note: an automata can be described by Kripke Structures.

Example 1 *:*

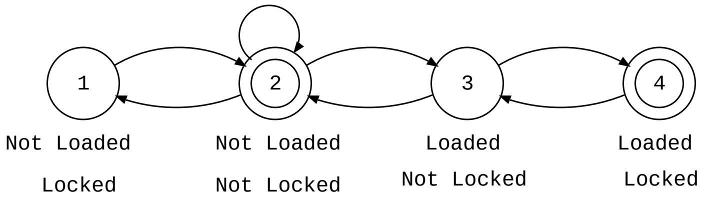

Figure 1: An example of Kripke Structure

In the above example

$W = \{1, 2, 3, 4\}$,
$W_0=\{2, 4\}$,
$T=\{(1, 2), (2, 1), (2, 3), (3, 2), (3, 4), (4, 3), (2, 2)\}$,
$P=\{Locked, Loaded\}$
$L(1) = \{Locked\}$, $L(2) = \phi$, $L(3)=\{Loaded\}$, $L(4)= \{Locked, Loaded\}=P$

See figure 1. A example of a path is: 2 3 4 . . .
Another path is 2 2 2 . . .

2 CTL

Consider a set of atomic propositions P, a CTL formula is defined as follows:

- if p $\in$ P then p is a CTL–formula.
- if p, q are CTL–formula then:
  - $\neg$ p, p $\vee$ q are also CTL–formula.
  - AXp and EXp are CTL–formula.
  - A(pUq) and E(pUq) are CTL–formula.

where

- A: means always (i.e. $\forall$ paths)
- E: means Eventually (i.e. $\exists$ a path).
- X: means successor (i.e. next moment of time)
- U: means until (i.e. holds until something) e.g. pUq means p is true until q is true.

Some CTL–fromulas that are syntatically correct in example 1 are:

1. AX(Loaded) $\vee$ ($\neg$Loaded).

2. A(Loaded U Locked) means: $\forall$ paths *Loaded* holds until *Locked* is true.

3. EX(Loaded $\wedge$ Locked) means: $\exists$ a path where Loaded and Loacked holds in the next moment of time.

The semantic of a CTL–formula is given with respect to a Kriple structure K.

Definition 1 *K, $w \models p$, where $w \in W$, p is a CTL–formula, means p holds in state w. It is defined as follows.*

- *If $p \in P$, K, $w \models p$ iff $p \in L(W)$.*
- *if K, $w \models \neg\, p$ iff K,$w \not\models p$.*
- *if K, $w \models p \vee q$ iff K, $w \models p$ or K, $w \models q$.*
- *K, $w \models AXp$ iff $\forall$ paths $w_0 w_1 \ldots$ with w=w_0, K, $w_1 \models p$*
- *K, $w \models EXp$ iff $\exists$ a path $w_0 w_1 \ldots$ with w=w_0, K, $w_1 \models p$*
- *K, $w \models A(p\ U\ q)$ iff $\forall$ paths $w_0 w_1 \ldots$ where (w_0=w) $\exists\ i \geq 0 : K, w_i \models q$ and K, $w_j \models p$ where $0 \leq j < i$.*
- *K, $w \models E(p\ U\ q)$ iff $\exists$ paths $w_0 w_1 \ldots$ where (w_0=w) $\exists\ i \geq 0 : K, w_i \models q$ and K, $w_j \models p$ where $0 \leq j < i$.*

Example 2 *Consider example 1, E(trueU Loaded) means: $\exists$ a path in which Loaded is true, path like 4 ....*
So: K, $\{4\} \models$ (true U Loaded) holds, and K, $\{4\} \models$ A(true U Loaded) also hold.

Example 3 *Consider example 1, So: K, $\{2\} \models$ A(true U Loaded) does not hold since taking the path: 2 2 2 ... does not satisfies CTL–formula.*

Definition 2 *$K \models p$ iff K, $w \models p$ for all $w \in W_0$.*

Notations:
A(true$\cup$ q) is often denoted by AFq means $\forall$ paths q will be true in the future.
E(true$\cup$ q) is often denoted by EFq means $\exists$ a path where q will be true in the future.

Definition 3 *Model checking problem is the problem of determining whether K $\models$ p holds given K and p.*

Question: How can we do that?

Consider the previous example 1, for p = AF Loaded, p does not holds in K, because we can find a path 2 2 2 ... such that K, 2 $\not\models$ p.
But for p=EF Loaded, p holds in K.
For p=EF(Loaded $\wedge$ Locked), p holds in K.

For p=EF(Loaded $\wedge\neg$ Loaded), p does not hold in K, because there is no state in the structure such that Loaded $\wedge\neg$ Loaded holds.

An algorithm to return true/false for the formula EFp where p is a propositional formula with respect to K is called McheckEF(K, p) as shown below:

```
1 function Mcheck(K, p)
2     CS = φ
3     NS = STATES(p, K)
4     while (NS ≠ CS) do
5           if (W0 ⊆ NS) then return True
6           CS = NS
7           NS = NS ∪ OneStepMcheck(NS, K)
8     endwhile
9     return false
```

The algorithm starts from p and builds the set of states NS backward. The function STATES(p, K) will return all states such that p holds (line 3). I.e.,

$$\mathrm{STATES(p, K)} = \{\mathrm{s} \in \mathrm{W} : \mathrm{p} \in \mathrm{L(s)}\}$$

The function OneStepMcheck(NS, K) will return all states that have at lease one immediate successor state in NS:

$$OneStepMcheck(NS, K) = \{s \in W : \exists s'(s' \in NS \wedge T(s, s'))\}$$

Example 4 *Consider the previous example 1. We want to prove the CTL formula:*

$$\mathrm{EF(Loaded} \wedge \mathrm{Locked)}$$

Applying the algorithm Mcheck(K, p), in line 3 we get NS=\{4\}. In line 5, $W_0 \not\subseteq$ NS. Therefore, continue to line 7, NS=\{3, 4\}, where state 3 is the immediate previous state of 4. returning to line 5, still $W_0 \not\subseteq$ NS. Then, at line 7 again, OneStepMcheck(\{3, 4\}, K) returns \{2, 4\}, but since 4 already exist, then the algorithm will add onle 2 to the set NS (i.e. NS=\{2, 3, 4\}). Now $W_0 \subseteq$ NS, so the algorithm returns true and stop.

3 Planning Domain

To relate the model checking problem to the problem of generating a plan, we need to define the planning problem as <D, I, G> where :

- D is a planning domain defined below.
- I is the set of initial states (we assume for now that I = $\{s_0\}$ which means I is only a set of one state).

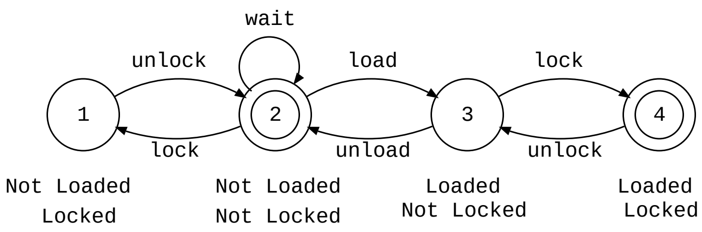

Figure 2: System description

- G is the goal states, we allow the goal to be conjuction of propositions.

a planning domain is defined by 4–elements: D=<F, S, A, R> where:

- F is a set of fluents.
- S is a set of states (S $\subseteq 2^F$).
- A is a set of actions.
- R is a function: R: S $\times$ A $\rightarrow$ S. Note that R is a function which means that there is one state for R(s, a) : s $\in$ S and a $\in$ A.

Note that if we have a planning problem represented by STRIPS, we can easily transfer it to the planning domain. i.e. for every (s, a) if s $\models Pre_a$ then:

$$R(s,a) = s \cup Add_a \backslash Del_a$$

Then, we can construct Kripke Structure: K = <W, W_0, T, L> where

- W = S.
- W_0 = I.
- for each a and s such that R(s, a) = s' then (s, s') $\in$ T.
- L(s)=s.
- P=F.

Example 5 *Consider the previous example 1:*

- $F = \{$*Loaded, Locked* $\}$
- $A = \{$*Lock, Load, wait, Unlock, Unload* $\}$
- $S = \{\phi$, $\{$*Locked*$\}$, $\{$*Loaded*$\}$, $\{$*Locked, Loaded* $\}$
- $R = \{$*(*s_0*, wait,* s_0*), (*s_0*, Loack,* s_1*), (*s_0*, Load,* s_2*), (*s_1*, Unlock,* s_0*), (*s_2*, Unload,* s_0*), (*s_2*, Lock,* s_3*), (*s_3*, Unlock,* s_2*)* $\}$

See figure 2

A plan π is a set of state–action pairs where:

$$\pi = \{(s, a) \mid s \in S \text{ and } a \in A\}$$

plan π is excutable if $\forall$ (s, a) $\in \pi$, R(s, a) $\neq \phi$.

Example 6 *Consider the previous example 5, we want to find a plan for the planning problem* <*D*, $\{2\}$, $\{4\}$> *where D defined in example 5. We can change algorithm MCheckEF(K, p) to return a plan instead of returning true or false. For example, we start with NS=*$\{4\}$ *and record* $\pi = \phi$*, then NS=*$\{3, 4\}$ *and* $\pi = \{$*(3, Lock)* $\}$. *After that, NS=*$\{2, 3, 4\}$ *and* $\pi = \{$*(2, Load), (3, Lock)* $\}$. *The plan is* $\pi = \{$*(2, Load), (3, Lock)* $\}$.