Title

Stefan Åhman
sahman@kth.se

Marcus Wallstersson
mwallst@kth.se

December 3, 2011

KTH Kista, Stockholm

Innehållsförteckning

1 Inledning

För att kunna kontrollera om en temporallogisk formel φ gäller i ett visst tillstånd s i en given modell $\mathcal{M}$ kan man använda sig av en modellprovare. Detta programverktyg måste i denna laboration implementeras att hantera följande delmängd CTL-reglerna (Computation tree logic):

$$\mathcal{M}, s \models \varphi$$

$$\phi ::= p \mid \neg p \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{AX}\ \phi \mid \mathsf{AG}\ \phi \mid \mathsf{EX}\ \phi \mid \mathsf{EG}\ \phi \mid \mathsf{EF}\ \phi$$

Modellen som ska kontrolleras kan beskrivas med en tillståndsgraf, där CTL används för att sätta upp villkor som måste uppfyllas av tillståndsgrafen samt tillstånden. Uppkomsten av oönskade stigar kan undvikas med specifika regler. Detta kan göras i denna laboration med bevissökning då bevissystemet som används är sunt och fullständigt och tillåter ändligt många bevisträd.

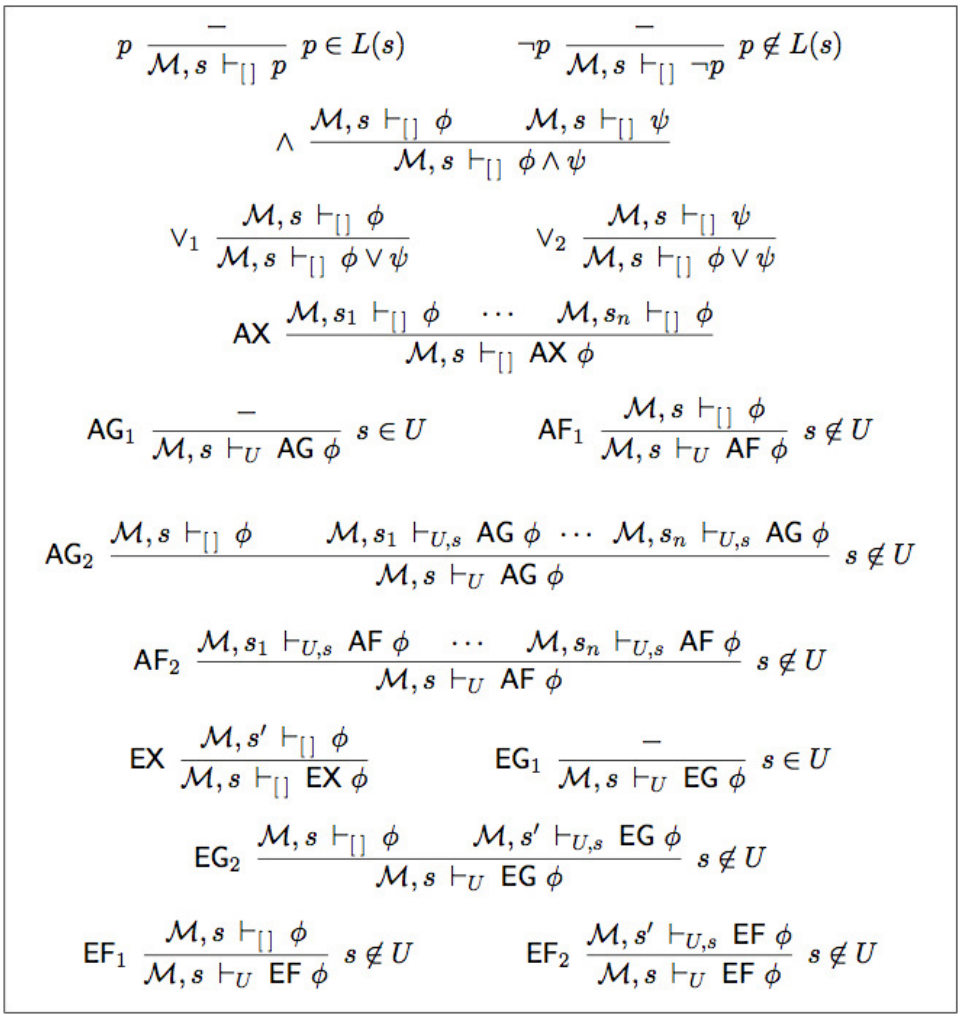
2 Problem och Syfte

Syftet med laborationsuppgiften är att:

- Fördjupa försåelsen för CTL och hur temporallogik kan användas för att specifiera viktiga systemegenskaper [HR04].
- Lära sig använda Prologs sökteknik för bevissökning.
- Lära sig bygga enkla men nyttiga programverktyg som kan användas till systemverifikation.

3 Genomförande

Modellprovaren skrevs i prolog då det är ett lämpligt programmeringsspråk för bevissökning. De befintliga reglerna för CTL implementerades. Vissa av reglerna kräver variabelt antal premisser och detta måste hanteras av programmet.



För att kunna testa modellprovaren fanns flertalet tester att tillgå som bestod av en liststruktur för att beskriva tillståndens egenskaper och grannar, detta beskrivs tydligare under Modell. Programmet skrevs så att en funktion "check" anropades med följande inparametrar:

```
check(T, L, S, U, F)
T - Alla tillstånd och dess grannar i listform
L - Lista över egenskaper i varje tillstånd
S - Aktuellt tillstånd
U - Lista för besökta tillstånd
F - CTL formel som ska testas
```

Check skrevs så att den med pattern matching kan matchas mot alla de regler som skulle implementeras. De matchades på följande sätt: `X`, `neg(X)`, `and(F,G)`, `or(F,G)`, `ax(X)`, `ag(X)`, `ex(X)`, `eg(X)`, `ef(X)`.

Nedan följer ett utrag ur programkoden för kontroll av `ef(X)`:

```
% EF1:
check(T, L, S, U, ef(X)) :-
not(member(S,U)),
check(T, L, S, [], X).

% EF2:
check(T, L, S, U, ef(X)) :-
not(member(S,U)),
member([S,Srest],T),
echeck(T, L, Srest, [S|U], ef(X)).
```

Då check stötte på `ef(X)` försökte den först med implementationen EF1 och sedan om den evaluerades till false försökte den med EF2.

4 Resultat

Referenser

[HR04] Michael Huth och Mark Ryan (2004). *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press, second utgåvan.