Relatório do Exercício-Programa 2

MAC 0239 - Introdução à Lógica e Verificação de Programas

Gabriel Baptista - nUSP: 8941300
Helio Hideki Assakura Moreira - nUSP: 8941064

1 Introdução

O programa consiste na verificação simbólica de modelos, o qual deve-se implementar um verificador simbólico de modelos CTL(Computation Tree Logic), usando a biblioteca de BDDs(Binary Decision Diagrams) do python(Pyeda).O problema da Verificação de Modelos é: dado M, s e Φ, queremos verificar se M, s |= Φ devolvendo SIM, se essa relação for verdadeira ou NÃO, caso contrário, isto é, M, s |≠ Φ .

Um modelo M ,é dado pela tupla (S, →, L) (estrutura de Kripke). E é importante observar que para todo s ∈ S existe um s' ∈ S tal que (s,s') pertence a →, isto é, existe uma transição a partir de todo estado s ∈ S.

Para verificação de um certo modelo M, vimos 4 algoritmos de verificação dos mesmos em sala:): SAT, SATaf , SATex e SATeu ,os quais manipulam subconjuntos de estados de S e o modelo de transições de estados, dado pelas duplas s → s', com s, s' ∈ S .

Para tanto, os algoritmos SATex e SATeu utilizam o conceito de pré-imagem fraca. A pré-imagem fraca de um conjunto de estados X, denotada por Pre∃ (X), devolve o conjunto de todos os estados s ∈ S tal que, na estrutura de Kripke (modelo), existe uma aresta que sai de s e vai para algum estado s' ∈ X, ou seja: Pre∃(X) = {s ∈ S | ∃ s' , s → s' e s' ∈ X}, além dela , exits e a pré-imagem forte.A pré-imagem forte de um conjunto de estados X, denotada por Pre∀(X), devolve o conjunto de todos os estados s ∈ S tal que, na estrutura de Kripke (modelo), para todo s' ∈ S que exista uma aresta que sai de s e leve a s' então s' ∈ X, ou seja: Pre∀(X) = {s ∈ S | ∀s',s → s' implica que s' ∈ X} que é utilizada no algoritmo de SATaf.

Para facilitar a utilização da biblioteca dos BDDs e a verificação dos modelos,como vimos, um estado s é rotulado por um conjunto de proposições L(s), isto é L(s)={x1 , x2 , …, xm }, sendo m o número de proposições do modelo. Assim, representaremos um estado s como uma conjunção de xi , se xi ∈ L(s), e $\underline{xi}$, se xi ∈/ L(S). Por exemplo, sendo m=3 e L(s)= {x1 , x3 }, temos s dado pela fórmula lógica x1 ^ $\underline{x2}$ ^ x3 , representada pela função booleana:: Bs = x1 * $\underline{x2}$ * x3.

Seja BX a representação fatorada do conjunto de estados X e BX' , a representação fatorada do conjunto X' (isto é, envolvendo variáveis x' i e $\underline{x'i}$). Seja ainda o conjunto B→ a representação fatorada do modelo. Para computarmos de forma simbólica a pré-imagem fraca de um conjunto de estados X , dada pela Equação:Pre∃ (X) = {s ∈ S | ∃s' s → s' e s' ∈ X},temos que primeiro verificar quais são as relações s → s' que levam para estados em X, isto é X'. Para isso, basta fazer a conjunção BX' ^ B→ (BX' * B→). Note que na fórmula resultante são eliminados os fatores das disjunções em B→ que geram uma inconsistência

(p ∨¬ p') com as variáveis em BX' , uma vez que ⊥ ∨φ ≡ φ . FInalmente, usamos a operação Exists (vista na aula sobre BDDs) para "eliminar" todas as variáveis "linha" . Pre∃(X)= Exists (xi ') (BX' * B→) , com i=1, …,m .Assim, todas as operações dos algoritmos SAT, SATaf , SATex e SATeu , podem ser definidas de forma simbólica e podemos implementálas usando a biblioteca de BDDs PyEDA.

2 A lógica do Programa

Como pedido no enunciado do EP, a entrada do programa deve ser da seguinte forma(em 5 linhas):

```
N                                          // número de estados
[(n1,n2),(n3,n4), ...]                     // lista de pares, arestas do modelo de
Kripke
[("x1","x2", …),("x2", ... "x4"),...]      // lista com os rótulos dos N estados
AF EU(+(EG x1)(x2))(AF x1)                 // fórmula CTL a ser testada
k                                          // estado k de interesse
```

Ao ler da entrada, cria-se os caminhos que representam o grafo do modelo M utilizando a função c_graph, uma vez criados os caminhos, o programa cria 4 dicts que são:

(a) dict_bddvar: que contém as variáveis do BDD levando em consideração os estados
(b) dict_variables: que contém as fórmulas correspondentes a cada estado
(c) dict_bddvar_primed: que são as variáveis do estado futuro
(d) dict_formulas_primed:que têm as fórmulas do estados futuros

As funções cuja terminações são em primed fazem referência a observação encontrada na página 391 do livro 'Logic in Computer Science modelling and reasoning about systems' dos autores M. Huth e M. Ryan que diz: "Recall that the primed variables refer to the next state."

Então, o programa cria a fórmula das transições, com a função c_formula_transitions, podendo agora criar a fórmula de S que seria nada mais nada menos do que a disjunção das fórmulas dos estados, deixando o caminho livre para solucionar o problema da verificação do modelo em questão.

Está é a ideia geral envolvida em nosso programa, mas como descrito na introdução, os algoritmos SAT necessitam do cálculo da pré-imagem para de fato solucionar o problema.

A função da pré-imagem fraca, deve receber uma fórmula e passá-la para o estado futuro(substituindo o nome das variáveis para variáveis de estados futuros), então deve-se aplicar a operação and nesta nova fórmula em conjunto com a fórmula de transições e por último deve-se aplicar a operação exist,que como citada nos slides das aulas do monitor, é o equivalente a apply((+,restrict(1),restrict(0)), retornando assim a pré-imagem fraca do conjunto X em questão.

A função da pré-imagem forte,se faz uso da seguinte igualdade:

$Pre\forall(X) = \{s \in S \mid \forall s', s \rightarrow s' \text{ implica que } s' \in X\} = S - pre\exists(S - X)$, então ela faz uma operação com conjuntos e uma chamada à função da pré-imagem fraca com o parâmetro S-X.

Com as funções descritas acima, temos tudo que é necessário para aplicar os algoritmos SAT em questão e finalmente verificar o modelo entrado pelo usuário, para tal, utilizamos a função solutions, onde cria-se uma lista com todas as soluções das fórmulas dos estados,da seguinte maneira: dê um restrict(solução) no BDD recebido ao aplicar a função SAT, obtendo as soluções para a fórmula em questão, guarde essas soluções na lista. E então para verificar se o estado em questão satisfaz a fórmula, basta percorrer a lista e verificar se ele encontra-se nela ou não.

3 Conclusão

Com este EP, verificamos mais uma vez o quão útil BDDs podem ser, no EP1 os utilizamos para verificar o problema das n-rainhas, já neste, verificamos a validade de modelos CTL, onde o usuário deve entrar com uma série de comandos que representam o modelo em questão, a fórmula CTL e o estado de interesse e devolve todos os estados que satisfazem a fórmula e SAT ou UNSAT para o estado de interesse (verificação do mesmo).