6
Outras Abordagens para a Verificação Formal

6.1
Introdução

Nos capítulos anteriores deste trabalho, foi visto como tornar o cálculo de Hoare útil para a verificação de propriedades de um código de um programa. Desta forma, o mesmo pode ser útil para assegurar propriedades de segurança, tal como ocorre com a técnica de PCC. Nas seções seguintes, serão analisadas duas outras maneiras de se realizar verificação formal de programas, que são a álgebra de Kleene com testes e a lógica dinâmica, comparando-as com o cálculo de Hoare.

6.2
Álgebra de Kleene com Testes

6.2.1
Definição

A álgebra de Kleene (AK) [35] é a álgebra de expressões regulares. Ela é uma estrutura algébrica **(** K**, +, ., *, 0, 1)**, a qual é um semi-anel idempotente sob **+**, **.**, **0**, **1**, satisfazendo:

$$1 + pp^* = p^*$$

$$1 + p^*p = p^*$$

$$q + pr \leq r \rightarrow p^*q \leq r$$

$$p + rp \leq r \rightarrow qp^* \leq r$$

onde $\leq$ se refere a ordem parcial natural em K:

$$p \leq q \leftrightarrow p + q = q.$$

A operação de **+** dá o supremo com respeito a ordem natural $\leq$.

Já a álgebra de Kleene com testes (AKT) [35] é simplesmente uma álgebra de Kleene com uma sub-álgebra booleana embutida. Ou seja, ela é uma estrutura duplamente ordenada **(** K**, **B**, +, ., *, ¾, 0, 1)**, tal que:

- **(** K**, +, ., *, 0, 1)** é uma álgebra de Kleene;
- **(** B**, +, ., ¾, 0, 1)** é uma álgebra booleana; e
- $B \subseteq K$.

Os elementos de B são chamados de testes. A operação booleana de complemento ¾ é definida apenas em B. Já cada um dos operadores **+**, **.**, **0** e **1** possuem dois papéis: aplicados a elementos arbitrários de K, eles se referem as operações de escolha não-determinística, composição, **fail** e **skip**, respectivamente; aplicados a testes, eles recebem o significado adicional da disjunção booleana, conjunção, falsidade e verdade respectivamente.

Para aplicações de verificação de programas, a interpretação padrão seria uma álgebra de Kleene de relações binárias em um conjunto e a álgebra booleana de subconjuntos da relação de identidade. Pode-se também considerar modelos de traços, onde os elementos de Kleene são conjuntos de traços (seqüências de estados) e os elementos booleanos são conjuntos de estados (traços de tamanho zero).

6.2.2 AKT e Cálculo de Hoare Proposicional

Para comparar as duas abordagens, será utilizado o cálculo de Hoare proposicional (CHP), que consiste de proposições atômicas e símbolos de programas, os conectivos proposicionais usuais, as construções **if** e **while**, e as asserções de correção parcial (ACP) construídas a partir destes. Programas atômicos são interpretados como relações em um determinado conjunto, e as proposições atômicas são interpretadas como subconjuntos do mesmo. O sistema de dedução de CHP consiste das regras de composição, **while**, **if**, **skip**, as regras de enfraquecimento e fortalecimento, e lógica proposicional. A regra de atribuição é omitida, já que não há uma relação estrutural de primeira ordem em AKT sobre a qual seja possível interpretar as variáveis de um programa. Na prática, seu papel é desempenhado por ACPs sobre programas atômicos que são postulados como hipóteses [35].

Sabendo-se o que é o cálculo de Hoare proposicional, agora é possível codificar esta lógica em AKT e derivar a composição, o condicional, **while**, **skip** e as regras de enfraquecimento e fortalecimento como teoremas na álgebra de Kleene com testes [35].

Uma asserção de correção parcial **{ P } C { Q }** pode ser codificada em AKT pela equação:

$$PC\overline{Q} = 0.$$

Intuitivamente, ela diz que o programa **C** com a pré-condição **P** e a negação da pós-condição **Q** não tem execução que pare. Uma formulação equivalente é

$$PC = PCQ$$

a qual diz, intuitivamente, que testar **Q** depois de executar **PC** é sempre redundante. A equivalência das duas equações dadas acima pode ser facilmente provada utilizando a álgebra de Kleene com testes [35].

Dado o exposto, dá-se agora as equações que são equivalentes às regras do cálculo de Hoare proposicional, as quais podem ser provadas como teoremas de AKT:

- Composição:

$$(PC_1 = PC_1R) \wedge (RC_2 = RC_2Q) \rightarrow (PC_1C_2 = PC_1C_2Q)$$

- Condicional (**if-then-else**):

$$(BPC_1 = BPC_1Q) \wedge (\overline{B}PC_2 = \overline{B}PC_2Q) \rightarrow P(BC_1 + \overline{B}C_2) = P(BC_1 + \overline{B}C_2)Q$$

- Condicional (**if-then**): neste caso, é só substituir $\mathbf{C_2}$ por **1** (**skip**) e fazer as simplificações possíveis.

- **While**:

$$(BPC = BPCP) \rightarrow (P(BC)^*\overline{B} = P(BC)^*\overline{B}P)$$

- Fortalecimento da pré-condição:

$$(R \leq P) \wedge (PC = PCQ) \rightarrow (RC = RCQ)$$

- Enfraquecimento da pós-condição:

$$(PC = PCQ) \wedge (Q \leq R) \rightarrow (PC = PCR)$$

❖ **Skip**:

$$P1 = P1P$$

6.2.3
Comparação Entre as Duas Abordagens

A primeira vantagem de se utilizar o cálculo de Hoare ao invés da álgebra de Kleene com testes é a de que na primeira pode-se utilizar lógica de primeira ordem, o que torna a expressabilidade de propriedades da especificação do programa maior. A segunda é que as atribuições são tratadas diretamente, ao invés de ACPs sobre programas atômicos que são postulados como hipóteses. Mais ainda, o cálculo de Hoare parece ser uma abordagem mais natural, ou seja, mais fácil de ser entendida pelo ser humano, expressando mais claramente o significado de um trecho de código.

Uma vantagem de AKT sobre CHP, defendida em [35], é a de que, na última, as regras de inferência apenas aumentam o tamanho do programa, o que não é uma restrição em se tratando das equações da primeira abordagem. Portanto, a seguinte regra não é derivável no cálculo de Hoare, por exemplo:

$$\frac{\{ P \}\ \texttt{if B then C else C fi}\ \{ Q \}}{\{ P \}\ C\ \{ Q \}}$$

Todavia, este tipo de regra não parece ser útil para a correção formal de um programa, que é o tópico de interesse nesse trabalho, mas sim para a análise de equivalência entre programas.

6.3
Lógica Dinâmica

6.3.1
Definição

O papel pretendido pela lógica dinâmica é prover um formalismo natural para estudar asserções sobre programas, os quais são estruturas dinâmicas. Como é sabido que a lógica modal é um formalismo capaz de capturar situações

dinâmicas, a mesma será estendida para capturar o significado de programas, produzindo como resultado a lógica dinâmica [36].

Na lógica modal são estudadas uma coleção **W** de mundos possíveis ou estados, com uma relação r descrevendo a acessibilidade entre certos pares deles. Diz-se **[]p**, lendo-se "necessariamente **p**", para afirmar que em um dado estado **s** Î **W**, **p** é verdadeiro em todos os estados **t** acessíveis via r a partir de **s**. Dualmente, **<>p** ("possivelmente **p**") é verdadeiro em **s** se **p** é verdadeiro em pelo menos um estado acessível a partir de **s** [36].

O esquema **(W,** r **)** da lógica modal se encaixa bem para descrever programas. **W** pode ser o conjunto de todos os possíveis estados de execução relevantes aos programas considerados. A qualquer programa a pode-se associar uma relação de acessibilidade r**(** a **)** sobre **W** tal que **(s, t)** Î r**(** a **)** se e somente se **t** é um possível estado final de a com o estado inicial sendo **s**, ou seja, existe uma computação de a começando em **s** e terminando em **t**. Diz-se "possível" aqui porque é desejável considerar a noção mais geral de um programa não-determinístico [36].

Como a idéia abordada aqui gera muitas relações de acessibilidade, torna-se necessário qualificar as ocorrências de **[]** e **<>** para o r**(** a **)** particular ao qual se está referindo. Isto é feito simplesmente ao se colocar o programa a dentro da modalidade. Então, tem-se: **[**a**]** e **<**a**>** [36].

6.3.2
Comparação Entre o Cálculo de Hoare e a Lógica Dinâmica

Considerando-se apenas programas determinísticos, para cada estado inicial só haverá no máximo um estado final se o programa parar. Se o estado inicial for a pré-condição **P** e o final, a pós-condição **Q** da asserção **{ P } C { Q }** do cálculo de Hoare, pode-se expressá-la em lógica dinâmica da seguinte forma:

$$P \rightarrow [C]Q.$$

Deve ser usado o "necessariamente", já que não se pode garantir que há um estado final (o programa pode não parar) e o cálculo de Hoare não tem poder para expressar parada de programas.

Note-se que se o "necessariamente" fosse substituído por "possivelmente", a sentença acima seria válida somente se o programa parasse, já que deve

haver pelo menos um estado onde **Q** vale. Esta é uma vantagem da lógica dinâmica sobre o cálculo de Hoare, que é a possibilidade de expressar a parada de programas. Além disso, é possível se expressar vários tipos de propriedades a cerca da correção de programas mais complexas do que se pode expressar com o cálculo de Hoare.

Por outro lado, embora seja possível simular as regras do cálculo de Hoare em lógica dinâmica, criando, com isso, um sistema dedutivo para um subconjunto da mesma, não é possível criar um sistema dedutivo completo para esta lógica.

Portanto, para saber se uma sentença em lógica dinâmica a cerca do comportamento de um programa é válida ou não, utiliza-se a idéia de conseqüência lógica. Conseqüentemente, se a linguagem utilizada para descrever o programa for estendida, não é necessária a criação de novas regras de inferência, como ocorre com o cálculo de Hoare, o que é uma das vantagens desta lógica. Entretanto, o tratamento computacional para a interpretação semântica de uma sentença parece ser muito difícil.