2
Formas de Validação

Neste capítulo veremos alguns métodos de validação que são usualmente utilizados na literatura para a análise de sistemas concorrentes e distribuídos. Nesta descrição nos restringiremos a apresentar o funcionamento básico destes mecanismos.

2.1
Simulação

As ferramentas baseadas em simulação permitem a análise do comportamento de um sistema através de uma depuração das especificações. Assim como na depuração de código numa linguagem de programação, o aparecimento de um *bug* pode estar em função da entrada e/ou das escolhas feitas no caminho de execução do programa. Ou seja, é tarefa do testador encontrar tais entradas ou tal caminho de forma que o *bug* possa ser resolvido.

Outra forma de funcionamento dos simuladores é através de processos estocásticos. O objetivo é avaliar o desempenho do protocolo especificado sobre um cenário mais realístico. Entretanto, não é garantido que todos os estados possíveis de uma especificação serão visitados.

Entre os projetistas de protocolos e administradores de redes os simuladores (ns (BBE+99), GlomoSim (ZBG98) e MobiCS (RE01), por exemplo) são mais difundidos do que outras formas de validação. Estes ferramentas permitem uma especificação bem detalhada da rede, como topologia, largura de banda, tamanho de pacote, etc.

2.2
Verificação Formal

Diferentemente dos métodos de Simulação, nos métodos de verificação formal temos uma prova formal do que está sendo analisado. Diversas formas de verificação formal e combinações destas podem ser encontradas na literatura. Nesta seção listaremos apenas as mais relevantes para o processo verificação de protocolos.

2.2.1 Prova de Teoremas

Provadores de teoremas são programas de computador utilizados para a realização de provas de teoremas matemáticos. Como estes provadores tem um funcionamento complexo e podem resolver problemas muito difíceis, normalmente requerem interação com o usuário para que este oriente e simplifique o seu processo. Dependendo do grau de automação, o provador pode ser reduzido a um verificador de provas, onde o usuário fornece a prova de maneira formal e o processo se dá de maneira semi-automática.

Para a utilização destes provadores na verificação de protocolos, é necessário um esforço para a definição destes protocolos como conceitos matemáticos. Esta é uma tarefa árdua e demanda grande experiência por parte do testador (Rus97). Por exemplo, a linguagem de entrada destes provadores é uma lógica, cujas regras podem ser dadas pelo usuário do sistema. Comumente encontramos estes sistemas especificados em lógica de primeira ordem clássica, não-clássica ou de alta ordem.

Como exemplos de provadores de teoremas temos HOL (M.J88), ACL2 (KM94) e PVS (ORS92).

2.2.2 Verificação de Modelos

Verificação de modelos é uma técnica de verificação para sistemas de estados finitos onde, dadas uma especificação M e uma propriedade p, verifica se p é válida em M ($M \models p$). Diferentemente dos métodos de simulação, que analisam um traço de execução, estas ferramentas exploram exaustivamente toda a árvore de execução de uma especificação. Usualmente, quando uma propriedade não é validada numa verificação, o verificador retorna um contra-exemplo, que é um traço de execução que inicia no estado inicial da verificação e termina no estado em que a propriedade não foi verificada.

Esta verificação exaustiva da árvore de execução, apesar de dar mais garantia aos resultados retornados por estes verificadores, impõe limitações a este processo. Este limitação é comumente chamada de *Problema da Explosão de Estados*. Diversos algoritmos e estruturas tem sido propostos os quais tentam contornar esta limitação.

Apesar disso, este método é o mais utilizado para a verificação formal de protocolos. Esta preferência existe pois, na parte da especificação, as diversas linguagens dos verificadores existentes provêm facilidades de forma a simplificar o processo de especificação formal. Na parte da verificação, o processo é totalmente automático, bastando que o testador forneça a especificação e as

propriedades a serem verificadas. Além destas propriedades fornecidas pelo testador, os verificadores comumente detectam cenários errôneos como *deadlock*, existência de ciclos na especificação, porções da especificação inalcançáveis, etc.

Outra característica interessante de um verificador de modelos é sua aplicabilidade. Verificadores para sistemas de tempo-real (por exemplo UPPAAL (BLL+95) e KRONOS (DOTY96)) e probabilísticos (por exemplo PRISM (KNP01)) são alguns dos exemplos de verificadores a serem utilizados na análise de protocolos com restrição de tempo-real e probabilidade, respectivamente. Outros verificadores (NuSMV (CCGR00), Spin (Hol97), Murphi (Dil96), Mocha (AHM+98) e Maude (EMS03), por exemplo) diferem em outros aspectos como a linguagem de entrada, modularidade, composicionalidade no processo de verificação, algoritmos de verificação de modelos implementados e interface com o usuário.

Quanto à especificação das propriedades, estas são normalmente fornecidas através de fórmulas expressas numa lógica temporal como CTL e LTL (CGP00), por exemplo. Lógicas temporais diferem quanto ao poder de expressão, o que está diretamente relacionado à complexidade na verificação. Entretanto, a utilização destas linguagens para a definição de propriedades sobre um protocolo podem ser auxiliadas por padrões de especificação de propriedades de sistemas distribuídos, como os listados em (DAC98).

2.3 Nossa Visão

A técnica de simulação, apesar de não garantir formalmente o comportamento de um protocolo, tem grande importância para se ter uma noção inicial de que a especificação reflete a essência do protocolo. Tanto que a maioria dos ambientes de verificação formal possuem ferramentas de simulação agregadas para se ter esta garantia aparente.

Podemos comparar as técnicas de simulação e verificação de modelos como os testes de caixa branca e preta de Engenharia de Software, respectivamente. Na primeira, a análise é feita passo a passo, na especificação, enquanto que na segunda, a análise é na especificação como um todo. Apesar de, internamente, a verificação de modelos analisar cada estado possível de uma especificação, este processo não é interativo e só termina quando o verificador termina a verificação, sendo a propriedade validada ou não.

Um provador de teoremas, por sua vez, requer interação com usuário, de forma que a verificação possa ser simplificada. Diferentemente da verificação de modelos, um provador de teoremas permite a verificação de sistemas de

estados infinitos. Esta característica sugere uma combinação destas duas (2) técnicas, de forma a se ter o melhor dos dois (2) mundos: a usabilidade dos verificadores de modelos e o poder computacional dos provadores de teoremas (SAL (BGL+00) e PVS (ORS92), por exemplo).

Em todos os métodos de análise descritos, uma ligação crucial é a relação entre a especificação dada pelo usuário e a sua intenção. Ou seja, quão próximo está a especificação do real comportamento do protocolo. Uma especificação errônea pode levar o usuário a conclusões corretas quanto ao que foi especificado, mas incorretas quanto ao que se espera do protocolo.

Este problema da relação *especificação* × *intenção* pode ser abordado de algumas maneiras. Por exemplo, no caso de verificação de sistemas já implementados, podemos extrair o modelo do sistema à partir de sua implementação (JPF (HP00), por exemplo). Outra abordagem é a utilização de uma linguagem mais abstrata, de domínio específico, de forma a melhorar a legibilidade, manutenção e aumentar a confiança nas especificações. Nesta abordagem reside a proposta deste trabalho, a qual também é defendida através destes trabalhos (BHE04, BHE05a, BHE05b, BHE05c).