Metodos Formais em Engenharia de Software (2025/26)

SAT solvers & SMT solvers

O problema SAT (*Boolean Satisfiability Problem*) decide se uma fórmula lógica proposicional pode ser satisfeita, ou seja, se existe uma atribuição de valores às variáveis que torne a fórmula verdadeira. Embora o SAT seja um problema de decisão NP-completo, os *SAT solvers* são ferramentas que conseguem lidar com este problema de forma bastante eficaz. Os SAT solvers modernos conseguem dar resposta a fórmulas com centenas de milhares de variáveis e dezenas de milhões de cláusulas. Este tipo de ferramentas é muito útil, pois muitos problemas difíceis podem ser reduzidos à decisão da satisfazibilidade de fórmulas em lógica.

Existem várias técnicas e algoritmos para a resolução de SAT, e muitos SAT solvers disponíveis. Por exemplo: CryptoMiniSat, Lingeling, ou Glucose, entre outros. Normalmente, os SAT solvers recebem como entrada uma fórmula num formato sintático específico (*Conjunctive Normal Form*), sendo necessário primeiro transformar a fórmula de entrada para esse formato, preservando a satisfazibilidade. O formato DIMACS CNF é um formato textual normalizado de input/output para os SAT solvers. Mais informações em satlive.org e aqui.

O problema SMT (*Satisfiability Modulo Theories*) é o problema de satisfazibilidade para lógica de primeira ordem no âmbito de alguma teoria lógica específica - uma teoria lógica que fixa as interpretações de certos predicados e símbolos de função. Dito de outra forma, restringe-se a satisfazibilidade a uma classe específica de modelos, numa lógica de primeira ordem tipificada e com igualdade. Os *SMT solvers* são ferramentas que visam responder ao problema SMT. Como o problema não é decidível, pode ser necessário (ou conveniente) restringir a classe de fórmulas em consideração a um fragmento (isto é, restrição sintática) adequado.

Existem muitos SMT solvers disponíveis. Por exemplo: Z3, Alt-Ergo, MathSAT, cvc5, Yices2, entre outros. Alguns são direcionados a teorias específicas; muitos suportam o formato SMT-LIB (um formato textural normalizado de input/output para SMT solvers); muitos fornecem recursos não padronizados. Mais informação em smt-lib.org e aqui.

SAT solvers e SMT solvers são utilizados em diversas aplicações de software, abrangendo múltiplas áreas de trabalho. Os SAT solvers, por exemplo, são usados como backends nos próprios SMT solvers. A ferramenta Alloy (de modelação e análise formal de sistemas, baseado em lógica relacional), que vamos usar nas próximas aulas, usa o KodKod solver. A plataforma Why3 (para verificação dedutiva de programas), com que vamos trabalhar na parte final desta UC, depende de SMT solvers para resolver as condições de verificação (que são geradas para garantir que os programas satisfazem as especificações dos contratos).

Os SAT/SMT solvers estão amplamente disponíveis como software gratuito e de código aberto, e várias linguagens de programação oferecem bibliotecas que permitem que os programas comuniquem com os solvers de forma simples e integrada.

Nesta aula vamos usar a plataforma Colab para resolver os problemas dos exercícios da primeira aula, com o auxilio do popular SMT solver da Microsoft, Z3. Cada problema será lançado num Colab notebook, e será implementado em Python utilizando a biblioteca `z3-solver`. Embora os problemas estejam codificados em lógica proposicional e podessem ser resolvidos diretamente num SAT solver, vamos usar o Z3 utilizando apenas a teoria dos Booleanos.

Existe um pequeno tutorial do Z3Py, a biblioteca Python de interface para o Z3. Para nos familiarizarmos com a utilização desta biblioteca, vamos começar por ver uma implementação do problema de *feature model analysis* para o exemplo Mobile Phone, apresentado nas aulas. De seguida, propõe-se que completem os restantes notebooks, correspondentes aos exercícios lançados na ficha de modelação em lógica proposicional.

Colab notebooks:

- Mobile Phone (resolvido)
- Colocação de Paineis (Exercício 1)
- Distribuição de Gabinetes (Exercício 2)
- Alocação de Aulas (Exercício 3)
- Survey (Exercício 4)