

# Verifikace konečných systémů pomocí SMT-solveru

Martin Milata

Fakulta informatiky Masarykovy univerzity

23. června 2010

# Konečné systémy

- ▶ Konečný systém:
  - ▶ množina vrcholů (stavy)
  - ▶ množina hran (přechody)
  - ▶ podmnožina počátečních vrcholů
- ▶ Může reprezentovat jednoduchý program, hardwarový obvod, protokol, . . .
- ▶ Výčet stavů a přechodů bývá velký - proto se používá *implicitní reprezentace* KS.
- ▶ Příklad implicitní reprezentace: zdrojový kód programu.

## Dosažitelnost chybových stavů

- ▶ Některé stavy konečného systému můžeme považovat za chybové.
- ▶ Cíl: Pro (implicitně) zadaný konečný systém a zadaný popis chybových stavů rozhodnout, zda-li je z nějakého počátečního stavu možné dosáhnout nějaký chybový stav.
- ▶ Explicitní přístup: Z implicitní reprezentace vytvořit graf konečného systému a dosažitelnost rozhodnout pomocí grafového algoritmu (např. DFS).
- ▶ Problém: velikost grafu.

# SMT - satisfiability modulo theory

## SMT-solvery

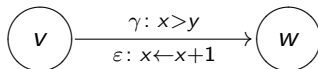
- ▶ Dokáží rozhodnout splnitelnost formule (fragmentu) predikátové logiky prvního řádu vzhledem k nějaké teorii.
- ▶ Příklad:  $x + 1 = 2 \cdot y \wedge x = y$ .
- ▶ Teorie: lineární aritmetika, pole, bitvektory, reálná čísla ...

## Základní myšlenka:

- ▶ Použít formule predikátové logiky pro reprezentaci množin stavů KS.
- ▶ SMT-solver pak dokáže rozhodnout, zda-li je tato množina prázdná.

# Algoritmus

Vstup: *Control flow graf* ověřovaného systému („program“),  
specifikace chybových stavů (funkce *assert*).



- ▶ Každému uzlu je přiřazena formule  $\varphi$ , která popisuje množinu špatných stavů. Stav je špatný, když
  - ▶ je chybový
  - ▶ se z něj lze dostat do chybového stavu
- ▶ Tyto formule se aktualizují podle vztahu

$$\varphi_v = \neg \text{assert}(v) \vee \bigvee_{(v,w)=e \in E} (\gamma(e) \wedge \text{wp}(\epsilon(e), \varphi_w)).$$

- ▶ Po dosažení pevného bodu kontrola, jestli formule počátečních stavů popisují prázdnou množinu.

## Implementace a experimenty

- ▶ Algoritmus implementován v nástroji dve-smt.
- ▶ Implementováno v jazyce Haskell.
- ▶ Vstupní formát nástroje DiVinE *DVE* - databáze Beem, více než 300 benchmarků.
- ▶ Vyhodnocení na těchto benchmarkcích negativní - v mnoha případech narůstají formule exponenciálně a SMT-solver je nevládne zpracovat.

Děkuji za pozornost.