Specifické techniky pro verifikaci softwarových systémů, abstraktní interpretace, metody abstrakce a aproximace, redukce částečným uspořádáním, metody zjemňování abstrakcí (např. CEGAR – protipříkladem řízené zjemňování abstrakcí).

1 Partial Order Reduction

- Idea: Z pohľadu LTL_X sú slová s vynechaným opakovaním ekvivalentné. Chceme teda zakázať niektoré prechody tak, aby sme odobrali len ekvivalentné slová.
- Defunujeme $ample(s) \subseteq enabled(s)$ tak, aby bol bezpečný.
- Prechody sú buď viditeľné alebo neviditeľné, podľa toho či menia množinu platných propozícií, alebo nie.
- Definujeme reláciu nezávislých prechodov a) Ak a aj b sú enabled a vykonám jedno z nich, to druhé je stále enabled. b) komutativita: $a(b(s)) = b(a(s))$.
- Potebujeme aby *ample* spĺňali tieto podmienky:
  - Ample je prázdny vtedy a len vtedy keď enabled je prázdny
  - Na všetkých cestách z s vykonávam len nezávislé prechody predtým ako vykonám ničo z ample. (Teda všetky cesty buď do nekonečna opakujú nezávislé prechody, alebo sa nakoniec objaví niečo z ample)
  - Ak s nie je celý rozvitý, tak všetko čo je v ample je invisible (inak povedané nemôžem len tak zahadzovať viditeľné zmeny)
  - Vo výslednej štruktúre nemôže byť cyklus ktorý obsahuje povolený prechod ktorý nie je v žiadnom ample na cykle.
- Ako toto prakticky počítať? 0 a 2 sú v pohode - lokálny check, 3ka sa zjednoduší na "na každom cykle je jeden fully expanded stav", takže ak sa to generuje cez DFS, stačí expandovať stav ktorý objaví stack. 2ka je rovnako ťažká ako reachabilita, takže ju len aproximujeme.
- Uvažujeme jednoduchý systém s paralelnými procesmi, shared variables a message queues. Ako kandidáta budem brať vždy možné prechody jednoho procesu. 0,2,3 kontrolujem lokálne (res. podľa DFS stacku), necessary condition pre 1ku je a) Neexistuje závislý prechod v inom procese (závislý = používa rovnakú frontu alebo premennú) b) Neexistuje lokálny (závislý) prechod, ktorý by som mohol uschopniť vykonaním prechodov iných procesov. Teda v žiadnom inom procese nie je prechod ktorý by mohol uschopniť niečo na mojom program counteri čo teraz nie je uschopnené ($pre(current_i(s) \setminus T_i(s)) \cap T_j = \emptyset$).

2 Abstraction

- Pridávam/Odoberám chovanie tak aby som dostal výrazne jednoduššší model.
- Partial order reduction, Cone of influence, Symmetry reduction, Equivalence reduction, etc.
- Stav programu sa nahradí abstraktným stavom - dva možné prístupy: Abstraktné domény alebo predikátová abstrakcia.
- Abstraktné domény: Zjednoduším domény premenných - -/0/+, mod, etc.
- Predikátová abstrakcia (presná): mám množinu predikátov, mapovanie stavov prebieha podľa platnosti predikátov. Problém: Nie každá množina stavov sa dá mapovať na jeden abstraktný stav — môže byť stále príliš komplikované.
- May abstrakcia (over) — ak existuje prechod v pôvodnej štruktúre, existuje prechod aj medzi abstrahovanými stavmi, môžu ale existovať aj falošné prechody.
- Must abstrakcia (under) — ak existuje prechod v pôvodnej štruktúre, pre každý vzor abstraktného stavu existuje prechod do vzoru cieľového stavu.
- Kartézska abstrakcia: Okrem 1/0 povolím ešte na predikátoch aj * — pre tento stav neviem. Pre ľubovoľnú množinu stavov potom viem zostrojiť abstraktný stav (čo najšpecifickejší).
- Príklad: Guarded command language - mám assignment u a guard g. Najskôr kontrolujem či náhodou nie je plnené $[b, \Phi] \implies \neg g$, v tom prípade žiadne prechody nemám. Inak v stave b môžem urobiť prechod do $b_i = 1$ ak viem dokázať, že $[b, \Phi] \implies (g \implies \varphi_i[x/u(x)])$ - inak povedané, ak som v stave b a splním guard, tak sa predikát stane platným po vykonaní assignmentu. Obodbne s $b_i = 0$ a $\neg\varphi_i$. Ak nič z toho neviem dokázať, musím ísť do $*$.
- Zjemňovanie abstrakcie - CEGAR - dostanem protipríklad. To je buď končený beh — V tom prípade ho proste potrebujem odsimulovať. Ak zistím že nemôžem pokračovať, musím zjemniť abstrakciu, teda pridať predikát ktorý rozlýši to čo som dosiahol od toho čo som nedosiahol. Ak je protipríklad laso, robím v podstate to isté, ale musím unrollovať aby som si bol istý. Unrollovať musím aspoň toľko krát, koľko je najmenšia mohutnosť stavu na lase (ak ten stav navštívim x-krát, muselo sa mi niečo zopakovať, lebo tam proste nie je viac stavov).
- Hľadanie najhrubšieho zjemnenia je NP-hard.

3 Abstraktná interpretácia

- POZOR NA COMPLETE LATTICE vs. COMPLETE PARTIAL ORDER!!!

- Statická analýza - zaujíma ma ktoré stavy odpovedajú miestam v programe, nezaujíma ma celý beh.
- Vety o pevnom bode pre complete lattice (Tarski - v complete lattice sú fixpointy zase complete lattice. Kleene - v complete lattice s final height je least fixed point vypočítaný ako $f(\emptyset)$ — pozn. complete lattice s final height je directed CPO, preto môžem použiť Kleeneho)
- Pre každú inštrukciu určím monotónnu funkciu v závislosti na tom, čo chcem počítať a spočítam fixed point.
- Ak to trvá príliš dlho, môžem použiť widening/narrowing (preskakujem rýchlejšie, res. si zmením trochu doménu aby som mohol dokonvergovať za cenu toho, že môžem prestreliť).