MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Post-mortem analýza stavového prostoru

BAKALÁŘSKÁ PRÁCE

Jan Kriho

Brno, jaro 2010

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Jan Kriho

Vedoucí práce: RNDr. Jiří Barnat Ph.D.

Shrnutí

Cílem této práce je doplnit verifikační nástroj DIVINE CLUSTER o možnost exportu při verifikaci vypočteného stavového prostoru do souboru a jeho předzpracování pro případnou budoucí post-mortem analýzu, např. extrakci jiného v době ukončení programu spočítaného protipříkladu.

Obsah

1 Úvod

Během 20. století se počítače postupně stávaly nedílnou součástí našich životů. Jejich přínos byl ohromný, a tak se nasazovaly do stále důležitějších oblastí našeho života. Do dnešního dne se rozšířily do takové míry, že dnes již najdeme jen málo přístrojů, které jimi nejsou řízeny. To však s sebou přineslo i několik negativních efektů.

Jak se již v minulosti mnohokrát potvrdilo, programování počítačového softwaru je proces náchylný k chybám. Některé problémy byly natolik závažné, že způsobily společnostem ohromné ztráty. Jedním z příkladů je raketa Ariane 5, která v roce 1996 krátce po startu explodovala kvůli chybě v konverzi 64bitového čísla na 16bitové, jak se tvrdí v [Lad02]. Další chyby v programech způsobily ztráty na životech – chyba v softwaru ozařovacího přístroje Therac-25 zapříčinila vystavení 6 pacientů nadměrné dávce radioaktivního záření, o čemž se píše v [Lev93].

Tyto a další události poukázaly na nutnost ověření funkce kódu před jeho vydáním, což vedlo k rozvoji různých validačních metod. Jedním z přístupů k řešení tohoto problému je tzv. formální verifikace, jejíž princip spočívá ve vygenerování modelu systému na základě kódu a ověření vlastností. Tento postup má však svá úskalí, jejichž jádro spočívá v nutnosti projít celý stavový prostor vygenerovaný modelem, což je mnohdy obrovské množství uzlů – tzv. „exploze stavů“

Distribuované verifikační prostředí DIVINE Cluster (z anglického Distributed Verification Environment), kterým se tato práce zabývá, řeší tento problém rozdělením práce na výpočetní cluster velkého množství strojů, které procházejí tento stavový prostor paralelně. Momentálně je DIVINE CLUSTER vyvíjen laboratoří PARADISE na Fakultě informatiky Masarykovy univerzity; poslední verze 0.8.3 vyšla 21. července 2009.

V současném stavu je DIVINE Cluster mimo jiné schopen provést formální verifikaci modelu proti určitým vlastnostem, ale chybí mu možnost zpětné analýzy proběhlého výpočtu. Úkolem této práce je připravit prostředí DIVINE Cluster pro uložení vypočteného stavového prostoru spolu s informacemi o něm na pevný disk ve formátu vhodném pro budoucí postmortem analýzu.

V průběhu zpracování stavového prostoru verifikačním algoritmem je ke každému vrcholu přiřazen tzv. apendix, do kterého si program ukládá data o průchodu tímto stavem. Vzhledem k dříve zmíněnému jevu stavové exploze mohou tato data velmi rychle zaplnit dostupnou operační paměť, tudíž je nutné tyto dvojice ukládat průběžně. Toho je možné dosáhnout

například spuštěním paralelního vlákna, které se bude starat o indexaci a organizaci dat v souboru a jejich kompresi. To vše je nutné provádět s ohledem na právě probíhající výpočet, jelikož spuštění paralelního vlákna a simultánní zápis na disk se může negativně odrazit na výkonu celého verifikačního procesu.

Popsaný problém lze částečně obejít tím, že na disk budou při běhu verifikačního algoritmu zapsána pouze surová data, která budou po skončení programu zpracována, rozdělena podle indexu stavu a zkomprimována. Případně může být provedena restrukturalizace uložení vrcholů v rámci celého clusteru pro co nejvyšší výkon v následném dotazování na jednotlivé stavy.

V této práci jsou implementovány oba způsoby ukládání stavového prostoru v průběhu výpočtu a do závěru je zahrnuto porovnání jejich efektivity a vliv na rychlost původního výpočtu.

Kapitola 2 popisuje komponenty distribuovaného verifikačního prostředí DIVINE a jazyk, který se v něm využívá. 3. kapitola se zabývá požadavky na efektivní systém pro ukládání jednotlivých stavů a jejich efektivní vyhledávání. Ve 4. kapitole lze nalézt zvolené implementace a analýzu jejch složitostí. Naleznete zde i důkazy pro korektnost výpočtu. V dodatcích se nachází příklady použité při testování implementací a grafické znázornění časové náročnosti při jejich provádění.

2 DiVinE

Tato kapitola je zaměřena na současný stav nástroje DIVINE Cluster, jeho komponenty a stručný popis použitých algoritmů a jazyků pro popis modelu.

2.1 Princip

Jak již bylo řečeno, DIVINE je distribuované prostředí určené k formální verifikaci modelů. Pro tento účel využívá sadu nástrojů pro generování Büchiho automatů, převod do jazyka DVE a následné generování stavů při jejich procházení zvoleným verifikačním algoritmem. Tyto prostředky jsou blíže popsány dále v této kapitole.

2.2 Lineární temporální logiky

Jako jazyk popisující požadované vlastnosti modelu se v systému DIVINE používá LTL, který je relativně přímočarý na vyhodnocení a dostatečně silný pro verifikaci programu. Jeho syntax vypadá následovně:

- $a \in \Sigma$ - atom jazyka
- $\phi = a \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \mathbf{X}\phi_1 \mid \phi_1 \mathbf{U} \phi_2$

Definice 1. *Mějme přechodový systém $T = (S, R)$, kde S je množina stavů a $R \subseteq S \times S$. Nechť $w \subseteq S$. Pak $w_1 = \{s' \in S \mid \exists s \in w : (s, s') \in R\}$. Analogicky $\forall n \in \mathbb{N} : w_{n+1} = \{s' \in S \mid \exists s \in w_n : (s, s') \in R\}$*

Sémantiku jazyka nyní lze vyjádřit následujícími tvrzeními:

- $w \models a \Leftrightarrow a \in w$
- $w \models \neg\phi \Leftrightarrow \neg\,(w \models \phi)$
- $w \models \phi_1 \wedge \phi_2 \Leftrightarrow (w \models \phi_1) \wedge (w \models \phi_2)$
- $w \models \mathbf{X}\phi \Leftrightarrow w_1 \models \phi$
- $w \models \phi_1 \mathbf{U} \phi_2 \Leftrightarrow \exists k \geq 0 : (w_k \models \phi_2 \wedge \forall i, 0 \leq i < k : w_i \models \phi_2)$

Další poznatky o LTL lze nalézt v [Eme95]

Literatura

[Lad02] LADKIN, P. B. *The Ariane 5 Accident: A Programming Problem?* [online]. Bielefeld: RVS Group, Posl. revize 24. 5. 2002 [cit. 15. 11. 2009].

URL <http://www.rvs.uni-bielefeld.de/publications/Reports/ariane.html>

[Lev93] LEVESON, N. *Medical Devices: The Therac 25* [online]. Seattle: University of Washington, 1993. [cit. 15. 11. 2009].

URL <http://www.cs.washington.edu/research/projects/safety/www/papers/therac.ps>

[Eme95] EMERSON, E. A. *Temporal and Modal Logic* [online]. Austin: University of Texas, 1995, s. 5–10.

URL <http://www.cs.utexas.edu/users/emerson/Pubs/handbook3.ps>