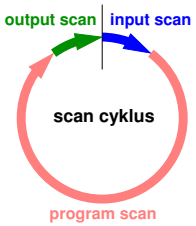
FORMÁLNÍ VERIFICKACE PLC PROGRAMŮ POMOCÍ SMV A UPPAAL

O. Šprdlík ** **R. Šusta** ***

Katedra řídicí techniky, Fakulta elektrotechnická ČVUT v Praze
Technická 2, 166 27 Praha 6
** *e-mail : sprdlo1@control.felk.cvut.cz, tel. : +420 2 2435 7682*
*** *e-mail : susta@control.felk.cvut.cz*

Abstrakt Formální verifikační techniky ověřují vlastností různých systémů, mimo jiné se mohou využít pro verifikaci programů programovatelných logických automatů (PLC). K provedení formální verifikace PLC programu je nutné jeho modelování ve verifikačním nástroji, například SMV nebo Uppaal. Článek předkládá postup modelování programu převedeného algoritmem APLCTRANS na soustavu logických rovnic, které tvoří abstrakci vhodnou pro modelování ve zmíněných nástrojích. Navržený postup dokáže modelovat pouze programy, které umí použitý algoritmus zpracovat, to znamená programy, které využívají omezených programovacích možností. Ukazuje se však, že výsledné modely mohou vést na nižší výpočetní nároky na proces verifikace než modely vytvořené obecnějšími postupy založenými na slučování modelů jednotlivých programových elementů (např. instrukcí). Postup se hodí pro modelování a verifikaci krátkých programů nebo dílčích bloků či funkcí rozsáhlejších řídicích algoritmů.

Klíčová slova: Programovatelný logický automat, verifikace, model checking



Obrázek 1. PLC scan cyklus

1. ÚVOD

cyklus

1.1 *Přístupy k modelování*

Existuje řada přístupů k verifikaci PLC programů. Většina z nich převádí do použitého verifikačního nástroje jednotlivé elementy programu jako jsou instrukce nebo příčky žebříčkových diagramů. Získané elementy se pak řadí sériově způsobem, který umožňuje použitý verifikační nástroj. Modelování jednotlivých instrukcí vytvoří poměrně přesný model programu, kde lze sledovat například hodnoty proměnných ve zvoleném místě programu. Tento přístup používají např. Canet et al. (2000) v SMV nebo Willems (1999) a Zoubek et al. (2003) v Uppaalu.

Proti detailnímu modelování stojí abstrakce, kdy provedení jednoho cyklu programu považujeme za nedělitelnou operaci. Šusta (2003) uvádí algoritmus APLCTRANS, který převádí program zapsaný v instrukcích na přiřazení určující hodnoty proměnných PLC po provedení cyklu programu jako funkce proměnných před provedením cyklu. Modely získané z této abstrakce mohou vést na nižší výpočetní nároky, určité srovnání výpočetních nároků verifikace SMV modelů s modely zís-

kaných z modelů instrukcí poskytuje Šprdlík and Šusta (2004).

1.2 *APLCTRANS*

Ústředním článkem převodu PLC programu abstrakcí z cyklu je automat, jehož jeden přechod odpovídá jednomu scan cyklu PLC. Můžeme ho definovat jako automat rodiny Mealy, potom jeho výstupům v daném okamžiku odpovídají hodnoty výstupů PLC po zkončení odpovídajícího scan cyklu. Přechodovou funkci automatu určují přiřazení hodnot určité podmnožině vnitřních a výstupních proměnných – těch, jejichž hodnota není konstantní a může mít přímý vliv na hodnoty některé jiné vnitřní nebo výstupní proměnné.

Přiřazení hodnoty proměnné nebudeme sledovat z pohledu instrukcí PLC, ale vně celého scan cyklu. Hodnotu proměnné po provedení jednoho cyklu programu určuje funkce hodnot proměnných před započetím cyklu. Pokud používáme pouze logické proměnné, můžeme tyto funkce zapsat pomocí logických výrazů. Šusta (2003) předkládá algoritmus APLCTRANS, který převádí program zapsaný v instrukcích *APLC programu*, které jsou obdobou instrukcí běžných v instrukčních sadách PLC různých výrobců, případně v normě IEC 1131-3. Výsledkem překladu je pak množina přiřazení logických funkcí proměnným PLC. Tato přiřazení nazýváme t-přiřazeními, pomocí nich můžeme definovat přechodovou funkci výše zmíněného automatu, podrobněji viz Šusta (2003).

Například programu

```
Load x1
And x2
Store y
```

odpovídá přiřazení $y := x1 \wedge x2$. Algoritmus ale dokáže zpracovat i programy s vícenásobnými zápisy do proměnných, či s dopřednými skoky nepodmíněnými i podmíněnými.

2. POUŽITÉ VERIFIKAČNÍ NÁSTROJE

Program přeložený do množiny t-přiřazení budeme modelovat a verifikovat v nástrojích SMV a Uppaal. Oba patří mezi nejběžněni používané verifikační nástroje a jsou volně dostupné.

2.1 *SMV*

Symbolic Model Verifier (SMV, McMillan (1992)) umožňuje ověřování požadavků kladených na systémy s konečným počtem stavů v prostředí diskrétního času. Pro popis systémů SMV používá specifický modelovací jazyk umožňující modulární stavbu programu. Každý modul může pracovat se svými vstupními proměnnými (zadanými v hlavičce modulu) a obsahuje několik sekcí určených pro různé části popisu. Na obr. 2 můžeme vidět fragmenty modulu skládajícího se ze tří sekcí:

```
MODULE PLC(H_FLOW, L1_FAIL, L1_FAILURE, ...)
VAR
 M_L1_PRIO : boolean;
 M_L1_PROD : boolean;
 M_L2_PROD : boolean;
 M_osr1 : boolean;

ASSIGN
 init(M_L1_PRIO) := 0;
 init(M_L1_PROD) := 0;
 init(M_L2_PROD) := 0;
 init(M_osr1) := 0;
 next(M_L1_PRIO) := L1_PRIO;
 next(M_L1_PROD) := L1_PROD;
 next(M_L2_PROD) := L2_PROD;
 next(M_osr1) := osr1;

DEFINE
 ACT_L1 :=  !M_osr1 & line_swap & !M_L1_PRIO;
 ...
 L1_PRIO :=  !line_swap & M_L1_PRIO | M_osr1 ...;
 L1_PROD :=  L2_FAILURE & H_FLOW&!SP_FAIL & ...;
 L1_UP :=  L2_FAILURE & H_FLOW & !SP_FAIL & !...;
 ...
```

Obrázek 2. Modul SMV modelu

VAR obsahuje definice proměnných modulu. Na obrázku jsou proměnné `M_L1_PRIO`,... typu boolean, kromě toho je možné definovat celočíselné proměnné nebo proměnné s oborem hodnot daným výčtem.

ASSIGN přiřazuje proměnným hodnoty podle různých předpisů. Například přiřazení `init` udává počáteční hodnotu proměnné, `next` pak hodnotu proměnné v příštím okamžiku. Hodnota se nemusí zadat pouze jako konstanta (0 nebo 1 pro proměnné boolean), ale také zapsáním výrazu nad proměnnými modulu a jeho vstupy. Následující hodnota se pak určí ohodnocením výrazu momentálními hodnotami proměnných. Hodnotu výrazu můžeme přiřadit proměnné přímo, bez použití `next`, určíme tak její momentální hodnotu. Při použití přímých přiřazení je možný vznik algebraických smyček.

DEFINE definuje symbolická jména reprezentující výrazy. Na obr. 2 dole se nachází např. část definice symbolu `L1_PRIO`, který výše využívá přiřazení pro příští hodnotu `M_L1_PRIO`.

Moduly jsou pouze předlohy pro části modelu. Pro jeho vytvoření je třeba v souboru s popisem uvést hlavní modul programu a v něm případně definice vnořených instancí dalších modulů. Celý model pak vzniká jako paralelní kompozice všech vytvořených instancí modulů. K vnitřním proměnným instance modulu můžeme přistupovat pomocí tečkové notace (obr. 3 dole).

```
MODULE main
VAR
 H_FLOW : boolean;
 L1_FAIL : boolean;
 L1_FAILURE : boolean;
 L2_FAIL : boolean;
 L2_FAILURE : boolean;
 L_FLOW : boolean;
 SP_FAIL : boolean;
 line_swap : boolean;

 plc: PLC(H_FLOW, L1_FAIL, ..., line_swap);

SPEC
AG (L1_FAIL -> (!plc.L1_PUMP & !plc.L1_UP &
    !plc.L1_DOWN))
```

Obrázek 3. Hlavní modul

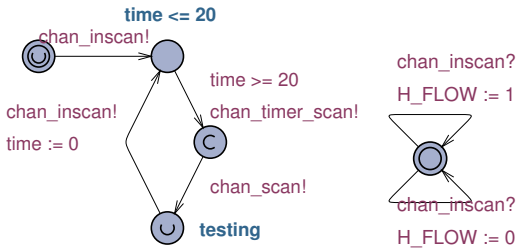
Dalšími sekcemi SMV modulů mohou m.j. být sekce SPEC – definice verifikovaných kritérií pomocí *Computation Tree Logic* (CTL, popsaná např. v McMillan (1992)). Jednu takovou definici obsahuje obr. 3: „Ve všech cestách vývoje stavu (operátor `A`) ve všech okamžicích (`G`) platí: Pokud je `L1_FAIL`, potom také platí výraz na pravé straně implikace (`->`).“

Systém SMV byl implementován vícekrát. Původní implementaci vytvořil K. McMillan, k verifikaci níže uvedeného příkladu byl použit NuSMV (IRST). Různé implementace se liší rozšířenými možnostmi modelování a verifikace, případně simulace, všechny ale umožňují prácí s modely zapsanými podle základní, zde uváděné syntaxe. Všemi je možné verifikovat model jednorázově, kdy se po proběhnutí verifikace zobrazí odpovědi, zda jsou jednotlivá kritéria splněná. V případě nesplnění se navíc vypíše sekvence stavů, která vede k němu vede.

2.2 *Uppaal*

Uppaal (Uppaal) je nástroj pro modelování, simulaci a verifikaci systémů reálného času vyvinutý univerzitami v Uppsale a Aalborgu. Narozdíl od SMV dokáže simulovat a verifikovat modely pracující s časem nejen jako kritériem uspořádání událostí, ale také z hlediska kvantitativního, tedy využívat hodnoty hodin při určení vývoje stavu či verifikovat časové požadavky na systém.

Model tvoří paralelní kompozice procesů – časových automatů. Časový automat zde nebudeme zavádět formálně, ale v krátkosti můžeme říci, že se jedná o množinu míst a přechodů mezi nimi. Pro definici časového automatu, ze kterého vychází Uppaal, viz Larsen (2002). V podmínce přechodu se může vyskytnout požadavek na čas, ve kterém může dojít k přechodu. S aktivací přechodu – změnou aktivního místa – může být spojen zápis do proměnné nebo hodin měřících čas. Dva paralelně běžící procesy ukazuje obr. 4.



Obrázek 4. Procesy v Uppaalu

Proces vlevo přejde v nulovém čase z počátečního místa přechodem znázorněným šipkou – v nulovém čase, protože se jedná o tzv. urgentní místo (označeno U). Poté se stav mění cyklicky. Délka cyklu je nastavena na 20 jednotek tím, že během aktivity místa následujícím za počátečním může být hodnota hodin měřících čas (`time`) nejvýše rovna 20, podmínka přechodu z tohoto místa je `time >= 20`, v dalších dvou místech se stejně jako v počátečním nevyvíjí čas (význam místa typu *committed* – označeno C – je podobný jako v případě místa urgentního). Přechod z místa `testing` do místa s časovou podmínkou nuluje hodiny.

Oba procesy se synchronizují společným kanálem `chan_inscan`. Ve druhém procesu dojde k přechodu pouze v okamžiku aktivace příslušného přechodu v procesu prvním. Oba přechody druhého procesu nemají žádné podmínky, proto se vždy náhodně provede jeden z nich.

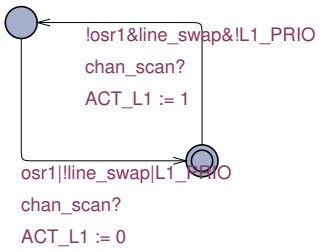
3. PŘEVOD DO SMV

Převod a jeho různé varianty zde nebudeme uvádět podrobně, zaměříme se pouze na hlavní myšlenky. Pro převod automatu generovaného APLC programem můžeme využít podobné sémantiky t-přiřazení a přiřazení `next`. Přepsáním všech t-přiřazení, které určují stav automatu popisujícího funkci PLC programu, do SMV jako přiřazení `next` získáme model vývoje stavu automatu. Proměnné PLC pak můžeme definovat jako symboly pro výrazy z příslušných t-přiřazení. Například programu

```
Load x1
And y1
Store y2
Load x2
And x3
Store y1
```

kde `xi` jsou vstupy a `yi` výstupy, odpovídají přiřazení $y1 := x2 \wedge x3$ a $y2 := x1 \wedge y1$. Mealyho automat pak má pouze dva stavy s přechodovou funkcí danou přiřazením pro $y1$. Na hodnotě $y2$ nezávisí přímo žádná jiná promenná, proto se její přiřazení neuplatní. Ústřední částí SMV programu pak bude

```
next(m_y1) := x2 & x3;
```



Obrázek 5. T-přiřazení v Uppaalu

```
DEFINE
 y1 := x2 & x3;
 y2 := x1 & m_y1;
```

Misto výrazu na pravé straně přiřazení `next` bychom mohli zapsat symbol `y1`.

4. PŘEVOD DO UPPAALU

4.1 *Převod automatu*

Pro převod automatu můžeme využít toho, že model v Uppaalu je paralelní kompozicí jednotlivých procesů. Model PLC programu může tvořit kompozice modelů jednotlivých t-přiřazení. Jednu z variant modelu t-přiřazení ukazuje obr. 5. Dvě místa procesu odpovídají hodnotám (0 a 1) proměnné `ACT_L1`, podmínky přechodů byly získány z t-přiřazení

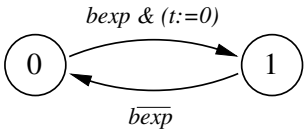
$$ACT\_L1 := line\_swap \wedge !osr1 \wedge !L1\_PRIO.$$

Všechny modely t-přiřazení synchronizuje pomocí společného kanálu (`chan_scan`) společný synchronizační proces (např. obr. 4 vlevo), v němž může být stanovena nebo omezena doba mezi dvěma přechody – trvání scan cyklu. Obr. 4 navíc obsahuje i model náhodně se měnícího vstupu PLC (vpravo).

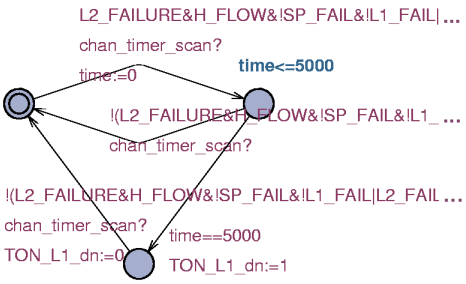
Mezi přechody pro synchronizaci modelů t-přiřazení (`chan_scan`) a modelů náhodných vstupních proměnných (`chan_inscan`) leží místo reprezentující konec scan cyklu – výstupy jsou již vypočtené, ale ještě se nenačetly nové hodnoty vstupů. Toto místo můžeme pojmenovat (na obr. 4 `testing`) a pak jeho jméno využít při deklaraci požadavků na systém, viz tab. 1.

4.2 *Modelování časovačů*

PLC programy pro svou funkci často využívají časovačů. Přidání možnosti modelovat časovače k pouhé práci s binárními proměnnými silně rozšiřuje množství případů, na které je možné využít daný postup modelování. Šusta (2003) navrhuje kompozici časovače typu *timer on delay* (TON) či *timer off delay* (TOFF) s automatem popisujícím



Obrázek 6. Automat pro časovač TON



Obrázek 7. Model časovače typu TON

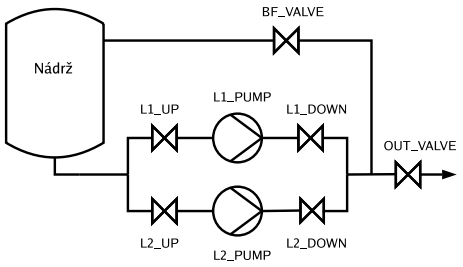
prácí PLC s proměnnými (odstavec 1.2). Před překladem programu pomocí APLCTRANS provedeme nahrazení zápisu do vstupu časovače (a jeho spuštění) zápisem do nově vytvořené výstupní proměnné symbolizující vstup časovače. Výstup časovače nahradíme vstupní proměnnou.

Časovač typu TON zpožďuje náběžnou hranu vstupu o určitou dobu, zpoždění. Tato doba bývá určena instrukcí spuštění časovače, případně jinou, obecně se ale může měnit. Obvykle se ale používají časovače s pevně zadaným zpožděním, APLC program samotné určení doby zpoždění neobsahuje. Na obr. 6 se nachází časový automat popisující funkci časovače TON. Symbol *bexp* reprezentuje výraz z t-přiřazení, které APLCTRANS vrátil pro proměnnou reprezentující vstup časovače. t jsou hodiny pro měření času. Výstup je určen výrazem

$$bexp \wedge (t > c),$$

kde c je zpoždění časovače. Výstup časovače můžeme zaznamenat do proměnné a tu pak využívat ve zbytku programu jako vstupní proměnnou. Model časovače pro Uppaal na obr. 7 používá jako výstup proměnnou `TON_L1_dn`, přechody mezi stavy podléhají podmínkám odvozeným z t-přiřazení pro proměnnou symbolizující v APLC programu vstup časovače. Zpoždění je pevně nastavené na 5000 časových jednotek.

Předpokládejme, že v původním PLC programu vstup časovače předchází všem čtením jeho výstupu, potom je hodnota výstupu v době čtení již ovlivněná hodnotou vstupu nastavenou v daném cyklu. V modelu odpovídající funkci zajistí synchronizace modelu časovače s přechodem, který předchází přechodu pro synchronizaci modelů t-přiřazení, na obr. 4 a 7 se tak děje kanálem `chan_timer_scan`. Případnému obrácenému pořadí vstupu časovače a čtení výstupu v programu bychom mohli model časovače synchronizovat se stejným přechodem jako modely t-přiřazení.



Obrázek 8. Schéma zapojení čerpadel

5. PŘÍKLAD VERIFIKACE

5.1 *Verifikovaný program*

Postupy převodu a verifikace byly aplikovány na program řízení čerpací jednotky se strukturou podle obr. 8 převzatý z Zoubek et al. (2003). Jednotka dodává vodu z nádrže pomocí dvou čerpadel. Ke každému čerpadlu je připojen přívodní a odtokový ventil, po jednom ventilu mají výtoková větev potrubí a větev zpětného toku.

Na řídicí systém jsou kladeny následující požadavky.

- *P1* Přítokový ventil čerpadla musí být otevřen alespoň 5 s před spuštěním čerpadla.
- *P2* Obě čerpací linky nesmí být nikdy spuštěné zároveň.
- *P3* Když je zastaven provoz, musí být všechny akční členy vypnuty (čerpadla zastavená, ventily zavřené).
- *P4* V případě poruchy čerpadla budou všechny související akční členy vypnuty.

Program přistupuje k řízenému procesu pomocí vstupů a výstupů PLC. V obr. 8 jsou u jednotlivých akčních členů zaznamenána jména proměnných zastupujících příslušné výstupy. Logická hodnota 1 na výstupu pro ventil znamená otevření ventilu, jednička na výstupu pro čerpadlo jeho spuštění. Program využívá řadu vstupů, patří k nim např. signalizace velikosti požadavku na odběr (L_FLOW, H_FLOW), signalizace poruchy čerpadla (L*i*_FAIL) nebo zastavení provozu (SP_FAIL).

Zoubek et al. (2003) ukazují verifikaci řídicího programu zapsaného v žebříčkovém diagramu. Obr. 9 obsahuje pouze jeho přepis do APLC programu. Bloky kódu na obrázku označené čísly odpovídají jednotlivým částem žebříčkového diagramu. Vstupy časovačů jsou nahrazeny proměnnými TON_L*i*_in. REdge je instrukce pro detekci náběžné hrany.

5.2 *Modelování programu a zápis požadavků*

Program můžeme přeložit pomocí APLCTRANSu a poté ho modelovat výše naznačeným postupem jako model v nástroji Uppaal. Formule *P2–4* lze v Uppaalu zapsat přímo – kromě proměnných vázajících se k požadavku se ve formuli bude vyskytovat pouze indikátor přítomnosti synchronizačního procesu v místě po provedení programu (`testing` na obr. 4).

Požadavek *P1* nelze zapsat přímo. K jeho ověření je třeba přidat do modelu hodiny, které budou měřit čas od otevření přítokového ventilu. Nové hodiny mohou být nulovány v každém cyklu při L*i*_UP=0 a do formule se pak vloží požadavek, že L*i*_PUMP nesmí být roven 1 pokud je hodnota hodin menší než 5000 – tuto variantu používá Zoubek et al. (2003). Jinou možností, která nevyžaduje nulování hodin v každém cyklu, je provedení jejich resetu pouze na hraně vedoucí z místa symbolizujícího logickou 0 proměnné L*i*_UP do místa pro 1 v procesu příslušejícím této proměnné. Požadavek potom pro čerpadlo L1 vyjadřuje formule pro *P1* z tab. 1. V tabulce se nacházejí specifikace dalších požadavků – *P4* také pouze pro L1. Operátor `A[]` znamená „vždy ve všech cestách vývoje stavu“, vykřičník představuje negaci.

Zoubek et al. (2003) konstatují, že se jim z důvodů vysoké paměťové náročnosti nepodařilo provést verifikaci celého programu modelovaného jejich nástrojem. Proto používají abstrakce založené na odstranění některých příček z programu před jeho modelováním:

Požadavek *P1* závisí na příčkách 8–10. Pokud z programu ostatní příčky odstraníme a budeme proměnnou L1_PROD uvažovat jako vstup s náhodně se měnící hodnotou, ověříme modelovanou část programu proti všem možným sekvencím této

```
0  Load line_swap
   REdge osr1
   Push
   And !L1_PRIO
   Store ACT_L1
   Pop
   And L1_PRIO
   Store ACT_L2
1  Load ACT_L1
   Or L1_PRIO
   And !ACT_L2
   Store L1_PRIO
2  Load L1_PRIO
   And !L2_PROD
   Or L2_FAILURE
   Store RS_L1_S
3  Load !H_FLOW
   And !L_FLOW
   Or L1_FAIL
   Or SP_FAIL
   Store RS_L1_R1
4  Load RS_L1_S
   Or L1_PROD
   And !RS_L1_R1
   Store L1_PROD
5  Load !L1_PRIO
   Or L1_FAILURE
   Store RS_L2_S
6  Load !H_FLOW
   And !L_FLOW
   Or L2_FAIL
   Or SP_FAIL
   Or L1_PROD
   Store RS_L2_R1
7  Load RS_L2_S
   Or L2_PROD
   And !RS_L2_R1
   Store L2_PROD
8  Load L1_PROD
   Store L1_UP
9  Load L1_PROD
   Store TON_L1_in
10 Load TON_L1_dn
   Store L1_PUMP
   Store L1_DOWN
11 Load L2_PROD
   Store L2_UP
12 Load L2_PROD
   Store TON_L2_in
13 Load TON_L2_dn
   Store L2_PUMP
   Store L2_DOWN
14 Load L1_PROD
   Or L2_PROD
   Store
   OUT_VALVE
15 Load L1_PUMP
   Or L2_PUMP
   And !H_FLOW
   Store BF_VALVE
   End
```

Obrázek 9. Program řízení čerpací jednotky zapsaný v instrukcích APLC

P1	`A[] !(proc_cycle.testing and (_proc_L1_UP.time<5000 or !L1_UP) and L1_PUMP)`
P2	`A[] !(proc_cycle.testing and L1_UP and L1_PUMP and L1_DOWN and L2_UP and L2_PUMP and L2_DOWN)`
P3	`A[] (proc_cycle.testing and SP_FAIL) imply (!L1_PUMP and !L1_UP and !L1_DOWN and !L2_PUMP and !L2_UP and !L2_DOWN and !OUT_VALVE and !BF_VALVE)`
P4	`A[] (proc_cycle.testing and L1_FAIL) imply (!L1_PUMP and !L1_UP and !L1_DOWN)`

Tabulka 1. Požadavky jako formule pro Uppaal

proměnné, tedy i proti těm, které mohou nastat v celém programu.

Požadavky *P2–4* nezávisejí přímo na čase, lze je ověřit při odebrání časovačů: Příčky 9–10 a 12–13 nahradíme přímým zápisem hodnoty proměnné L1_PROD do výstupů L1_PUMP a L1_DOWN resp. zápisem L2_PROD do výstupů L2_PUMP a L2_DOWN.

5.3 *Výsledky verifikace*

Použitý program splňuje všechny formule z tab. 1. K tomuto výsledku dospěla jak verifikace popsaná v Zoubek et al. (2003), tak i verifikace, provedená na modelech získaných postupy z předchozích odstavců. Verifikace proběhla na počítači s procesorem AMD K6-III, 450MHz a 192MB RAM.

Časová náročnost verifikace jednotlivých formulí na různých modelech se výrazně liší, jak ukazuje tab. 2. Její sloupce odpovídají různým postupům modelování. Všechny modely, které popisují časové chování cyklu, předpokládají jeho pevnou délku 20 ms. V některých případech byla uplatněna „Cone of Influence Reduction“ (COI redukce), tj. odstranění modelů t-přiřazení, které nemají vliv na splnění verifikované formule.

- M1 obsahuje časy verifikace modelů získaných z celého programu jeho překladem na transmnožinu. Modely byly redukovány COI redukcí pro proměnné obsažené ve verifikované formuli.
- M2 představuje modely využívající abstrakcí podle Zoubek et al. (2003) bez použití COI redukce.
- M3 ∼M2, ale s COI redukcí.
- M4 ∼M3, ale modely nepopisují časové trvání cyklu, doba celého jejich cyklu je nulová.
- SMV obsahuje časy verifikace formulí na modelech získaných použitím stejných abstrakcí jako v M2. Tentokrát ale modely byly vytvořené pro SMV jako automaty typu Mealy a verifikovány pomocí NuSMV.

	M1	M2	M3	M4	SMV
P1	43:55	0,32	0,3	–	–
P2	20:11	37,2	11,4	4,8	0,09
P3	22:36	37,3	13,6	5,75	0,1
P4	8:14	37,2	9,0	4,16	0,09

Tabulka 2. Srovnání dob verifikace na různých modelech

Časy v tabulce jsou ve formátu „minuty ':' sekundy ',' neceločíselná část sekundy“.

Tabulka ukazuje, že při pouhém použití modelovacího postupu bez abstrakce provedené uživatelem trvá verifikace mnohonásobně delší dobu než při zjednodušení programu před jeho modelováním. Přesto je možné tento výsledek považovat za úspěch, protože se narozdíl od Zoubek et al. (2003) vůbec podařilo dosáhnout výsledku bez těchto abstrakcí.

Při použití převzatých abstrakcí programu (sloupec M2) se čas verifikace snížil. Sloupec M3 ukazuje vylepšení časů při použití COI redukce. Rychlost verifikace se dále zvýšila u programů bez použití časovačů a bez potřeby čas sledovat tím, že se vytvořil model s nulovou délkou cyklu (M4).

Výrazně nejmenších časových nároků verifikace dosáhl nečasový model pro SMV. Části SMV programu obsahují obr. 2 a 3.

6. ZÁVĚR

Představený převod usnadňuje využití dříve navrženého a implementovaného algoritmu APLC-TRANS (Šusta (2003)) k verifikaci PLC programů. Docílený postup verifikace se alespoň zčásti přibližuje ideálnímu stavu, kdy by se verifikace stala plně zautomatizovanou činností vyžadující minimum zásahů uživatele.

Mezi výhody použitého řešení patří možnost modelovat program obdobným postupem ve dvou odlišných nástrojích. Uppaal dokáže narozdíl od SMV modelovat a verifikovat časové charakteristiky systémů, v SMV je zase možné vyjádřit širší spektrum nečasových požadavků plnou CTL logikou.

REFERENCE

G. Canet, S. Couffin, J.J. Lesage, A. Petit, and P. Schnoebelen. Towards the automatic verification of PLC programs written in instruction list. In *IEEE Int. Conf. Systems, Man and Cybernetics*, 2000.

IRST. NuSMV: a new symbolic model checker [online]. URL `http://nusmv.irst.itc.it/`.

K. G. Larsen. UPPAAL implementation secrets. In *7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems*, 2002.

K.L. McMillan. *Symbolic Model Checking: An approach to the state explosion problem*. PhD thesis, Carnegie Mellon University, 1992.

Uppaal. [online]. URL `http://www.uppaal.com`.

H. X. Willems. Compact timed automata for PLC programs. Technical Report CSI-R9925, University of Nijmegen, 1999.

B. Zoubek, J.-M. Roussel, and M. Kwiatkowska. Towards automatic verification of ladder logic programs. In *IMACS Multiconference on Computational Engineering in Systems Applications (CESA)*, 2003.

O. Šprdlík and R. Šusta. SMV verification of PLC programs. In *6th International Scientific-Technical Conference on Process Control (Říp 2004)*, 2004.

R. Šusta. *Verification of PLC programs*. PhD thesis, Czech Technical University in Prague, Department of Cybernetics, 2003.