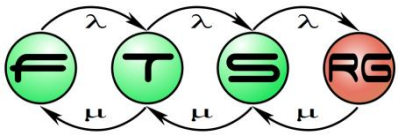
Petri-hálók elérhetőségi problémájának vizsgálata ellenpélda-alapú absztrakció finomítással (CEGAR)

Hajdu Ákos és
Mártonka Zoltán
munkája



Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

A feladat

- Adott egy Petri-háló és két állapot
- Kérdés: el tudunk-e jutni a kiinduló állapotból a cél állapotba engedélyezett állapotátmeneteken keresztül

Petri-háló elérhetőség

- A probléma eldönthető
  - Még nem adtak rá felső korlátot a komplexitásra
- Alsó korlát:
  - Legalább EXSPACE nehéz

Petri-háló elérhetőség

- Legtöbb eddigi megoldás elérhetőségi gráfon alapszik
- Elérhetőségi gráf:
  - kis háló esetén is lehet nagy
  - vagy akár végtelen
- Végtelen esetben a fedési gráfot tudjuk használni
  - Absztrakció
  - Véges reprezentáció

Petri-háló analízis

- Eldönthetetlen problémák:
  - 2 Petri-háló esetén az egyik elérhető állapotainak halmaza részhalmaza-e a másik háló elérhető állapotainak (subset problem)
  - 2 Petri-háló esetén az elérhető állapotok halmaza megegyezik-e (equality problem)
- Fedési probléma:
  - EXPSPACE nehéz

CEGAR megközelítés

- Absztrakció: állapotegyenlet
  - Lineáris egyenletrendszer, hatékonyan oldható meg
  - Felülbecsli a ténylegesen elérhető állapotokat
  - A lehetséges állapotok tere helyett az állapotegyenlet megoldásainak terét járjuk be
  - Minden megoldás felírható egy minimális megoldás és T-invariánsok lineáris kombinációjának összegeként

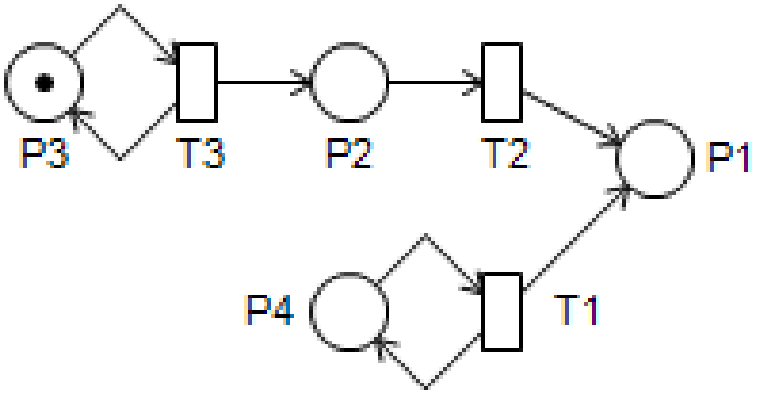
CEGAR megközelítés

- Ellenpélda alapú absztrakció finomítás
  - Megoldjuk az állapotegyenletet
    - Ha nincs megoldás, nem elérhető
    - Ha van, akkor meg kell vizsgálni, hogy eltüzelhető-e
      - Ha eltüzelhető -> elérhető
      - Ha nem tüzelhető el -> finomítani kell az absztrakción
  - Absztrakció finomítás
    - A nem eltüzelhető megoldásokat ki kell zárni, hogy újból megoldhassuk az állapotegyenletet
    - Ehhez lineáris egyenlőtlenségeket, ún. megkötéseket használunk

Megkötések

- 2 féle megkötést definiálunk, melyek tranzíciókra vonatkozó lineáris egyenlőtlenségek:
  - Jump megkötés
    - Alakja $t < n$
    - Elérhetjük, hogy egy másik minimális megoldást kapjunk meg
  - Increment megkötés
    - Alakja: $\Sigma n_i t_i \geq n$
    - Elérhetjük, hogy ne minimális megoldást kapjunk meg

Megkötések - példák

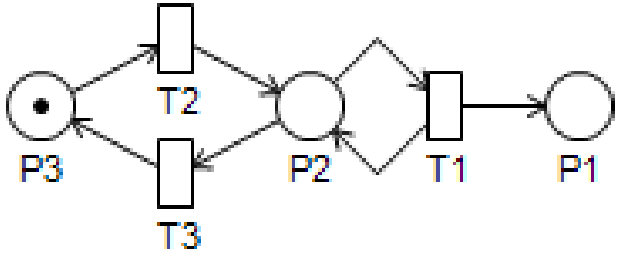


0 0 1 0 -> 1 0 1 0 elérhetőség

Minimális megoldás: 1 0 0 azaz T1 eltüzelése
Nem realizálható…

T1<1 megkötéssel kizárható
Új minimális megoldás: 0 1 1
Realizálható T3, T2 sorrendben

Megkötések - példák



0 0 1-> 1 0 1 elérhetőség

Minimális megoldás: 1 0 0 azaz T1 eltüzelése
Nem realizálható…

T2>=1 megkötéssel kizárható
Új, nem minimális megoldás: 1 1 1
Realizálható T2, T1, T3 sorrendben

Részleges megoldások

- Kérdés: hogyan készítsünk megkötéseket?
- Előtte szükség van a részleges megoldások fogalmára
- Részleges megoldás:
  - Négy elemből áll:
    - Eddigi megkötések listája (kezdetben üres)
    - Minimális megoldásvektor, ami kielégíti az állapotegyenletet és a megkötéseket is
    - Tüzelési sorozat, amiben minden tranzíció legfeljebb annyiszor tüzel, ahányszor a megoldás vektor szerint tüzelhet
    - Maradékvektor: megoldásvektor - Parikh(tüzelési sorozat)
  - A részleges megoldások tüzelési sorozatai maximálisak
  - Ha a maradékvektor nullvektor, teljes megoldásnak nevezzük

Részleges megoldások generálása

- Egy adott megkötéslistából és a hozzá tartozó megoldásvektorból hogyan lesz részleges megoldás?
- Brute force megoldás: Felépítünk egy fát, ahol a csúcsok markingok, az élek tranzíciók. Két csúcs között az él az oda eljutást jelöli, minden tranzíció legfeljebb annyiszor szerepelhet egy úton a gyökérből kiindulva, mint amennyiszer a megoldásvektorban szerepel
- A gyökérből minden levélbe vezető út egy maximális tüzelési sorozat, amihez tartozik egy új részleges megoldás
- Több helyen optimalizálható, később lesz róla szó

Megkötések készítése

- Jump megkötés
  - Ha egy új megoldásvektort kapunk, minden tranzícióhoz hozzáadunk egy megkötést, ha ezáltal kapunk megoldást, azt eltároljuk
- Increment megkötés
  - Ha egy részleges megoldás nem teljes az azt jelenti hogy elakadtunk
  - Megpróbálunk hozzávenni plusz tranzíciókat
  - 3 lépéses algoritmus

Megkötések készítése

- Increment megkötés (folyt.)
  - Felépítünk egy függőségi gráfot, amiből kiderül, hogy a tüzelni nem tudó tranzíciók mely helyek miatt nem tudnak tüzelni
  - Megbecsüljük hogy az előbb megkapott helyekre minimálisan hány token kell, hogy legyen esélyünk eltüzelni a kívánt tranzíciókat
  - Olyan tranzíciókat veszünk hozzá a megoldáshoz, amelyek az előbbi helyekre a kívánt mennyiségű tokent termelhetnek -> increment megkötés

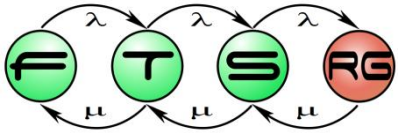
Jump megkötések átalakítása

- A jump és increment megkötések összeakadhatnak
  - Pl. t0<1 megkötéssel jutottunk el egy másik minimális megoldásba
  - Hozzáadunk egy invariánst, ami t0-n is átmegy
- Megoldás: jump megkötések átalakítása incrementté, mielőtt további increment megkötést adunk hozzá
  - Minden tranzícióra ami legalább egyszer eltüzel megkötés, hogy nem tüzelhessen kevesebbszer
  - Így nem ugorhatunk vissza korábbi minimális megoldásba

A teljes algoritmus (vázlatosan)

- A jump-okkal kapott minimális megoldásvektorokat egy listában tároljuk és sorban vesszük ki őket
- Részleges megoldásokat keresünk hozzájuk, amiket egy stack-ben tárolunk
- A részleges megoldásokhoz increment megkötést adva új megoldásvektort kapunk, amihez először jump-okkal keresünk más minimális megoldásokat, amiket a listába rakunk, utána folytatjuk ezzel az új vektorral
- Ha egy részleges megoldáshoz nem adható increment megkötés, vagy utána kielégíthetetlenné válik a rendszer, akkor előveszünk egy másik részleges megoldást a stack-ből
- Ha elfogynak a részleges megoldások előveszünk egy másik minimális megoldásvektort a listából

Optimalizációk



Optimalizációk

- Brute force fa építés optimalizálása
  - Részleges rendezés
  - Stubborn set
  - Különböző sorrendek összevonása
- Végtelen ciklusok kiküszöbölése
  - Ugyanazon invariáns többszöri hozzáadásának szűrése
- Részleges megoldások tárolása

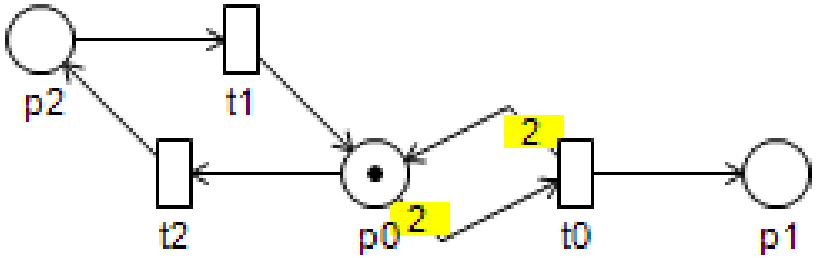
Brute force fa optimalizálása

- Stubborn set
  - Részleges megoldások keresésekor a fa méretét csökkentheti
  - Függőségek alapján meghatározza a tranzíciók egy halmazát, amit a többiek előtt eltüzelhetünk
  - Ez a halmaz legtöbbször sokkal kisebb mint az összes engedélyezett tranzíció
  - Kevesebb elágazás lesz a fában
- Különböző sorrendek összevonása
  - Ha ugyanazt a markingot többféle tranzíció tüzelési sorrenddel is elérjük, elég az egyik ágat megtartani
  - A továbbiakban úgyis csak az elért marking számít

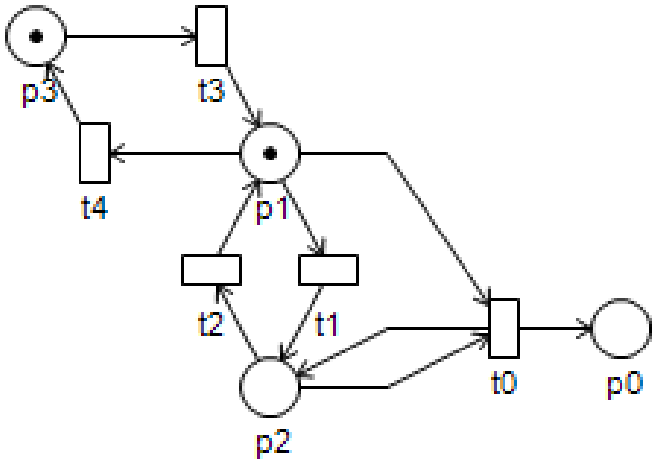
Megoldások szűrése

- Ha két részleges megoldás csak egy T invariánsban különbözik és azt el is tüzeltük, akkor nem jutottunk előrébb
- A T invariáns tüzelése közben előfordulhat, hogy valamely tüzelni nem tudó tranzíció engedélyezéséhez közelebb voltunk mint a végállapotban, ilyenkor ebből a közbülső állapotból folytatjuk

Megoldások szűrése - példa



1 0 0 -> 1 1 0 átmenet
T1, T2 invariáns szűrése

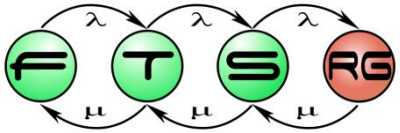


- 0 1 0 1 -> 1 1 0 1 átmenet
- T1, T2 invariáns szűrése
- De ha csak T1 tüzel javulás T0 szempontjából
- Innen folytatva hozzáveszi a T3, T4 invariánst

Megoldások tárolása

- A részleges megoldásokat eltároljuk, így elérhetjük, hogy ne dolgozzuk fel többször azokat amelyek több úton is elérhetőek
- Mi alapján hasonlítjuk össze a részleges megoldásokat?
  - Teljes egyezőség
    - Fölösleges megkötések szűrésével.
    - (Pl. t1>=3 és t1>=5 esetén az előbbi fölösleges)

Algoritmikus problémák és továbbfejlesztési lehetőségek

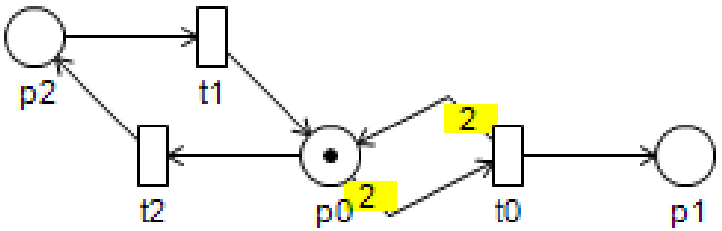


Rész-elérhetőségi probléma

- Nem egy konkrét állapot elérhetősége a kérdés
- Lineáris feltételek a helyekre vonatkozóan
- Átalakítható tranzíciókra vonatkozó feltételre
  - Helyekre vonatkozó feltétel, pl:
    - $a_1m_1+\ldots a_mm_n > b \Leftrightarrow \underline{a}\cdot\underline{m}>b$
  - Állapotegyenlet behelyettesítése
    - $m = m_0 + Cx$
    - $a(m_0+Cx)>b \Leftrightarrow (aC)x > b-am_0$
- Nem befolyásolja az algoritmus eddigi működését

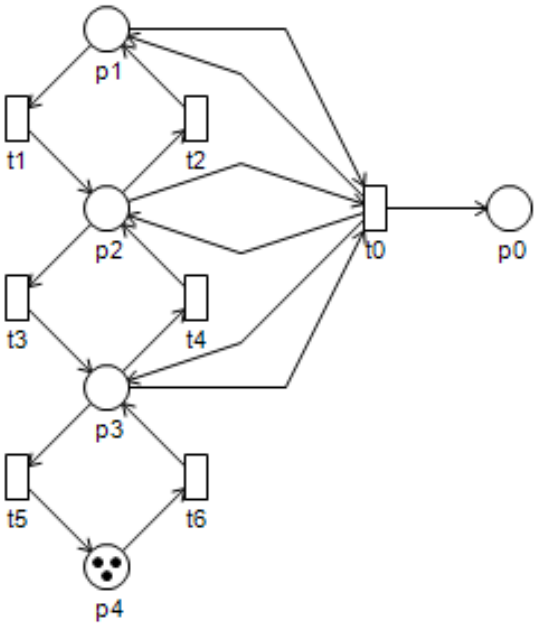
Megoldások szűrése

- Ha kiszűrünk egy megoldást nem garantálhatjuk a teljességet
- Néha már a szükséges feltétel is elég lenne annak belátására, hogy egy helyre nem kerülhet token -> szükséges feltétel gyorsteszt



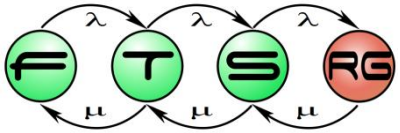
Megoldások szűrése

- Amikor visszalépünk keresni a jobb állapotokat, akkor már nem mindegy milyen sorrendben tüzeltek el a tranzíciók



- Először egy token megy körbe a T invariánsokon, nem elég…
- Utána két token -> nem mindegy a sorrend, mert lehet nem lesz javulás
- Megoldás: fa újraszámolása ilyen esetekben
- Lassít, de több esetre ad jó eredményt az algoritmus

Algoritmus kiterjesztése tiltó éles hálókra

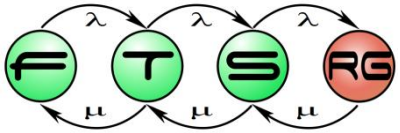


Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Tiltó éles hálók

- Turing gépekével azonos kifejezőerő
- Elérhetőség bizonyítottan nem eldönthető (>=2 tiltó él esetén)
- Alap algoritmus átalakítása
  - Helyekről tokeneket kell elvenni a tiltó él miatt
  - Increment megkötés, hasonlóan a token hozzáadáshoz, csak pont ellenkező céllal
- Optimalizációk
  - Stubborn set egyelőre nem támogatja a tiltó éleket

Fejlesztési lehetőségek



Megoldások szűrése

- Ha egy T invariánson nincs elég token, csináljunk!