九州大学学術情報リポジトリ
Kyushu University Institutional Repository

SSReflectを用いたペトリネットにおけるKarp-Miller加速の形式化

関根, 祥吾
千葉大学理学研究科

https://hdl.handle.net/2324/1520935

出版情報：MI lecture note series. 61, pp.1-8, 2015-03-06. 九州大学マス・フォア・インダストリ研究所
バージョン：
権利関係：

KYUSHU UNIVERSITY

SSReflectを用いたペトリネットにおける Karp-Miller加速の形式化

関根　祥吾

joint work with
山本 光晴（千葉大学）

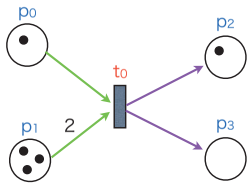
千葉大学理学研究科

December, 3, 2014

Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 1, 3, 1, 0\rangle}$

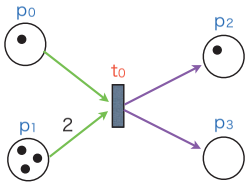


○ : プレース
▮ : トランジション
⇒ : アーク
• : トークン

Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 1, 3, 1, 0\rangle}$



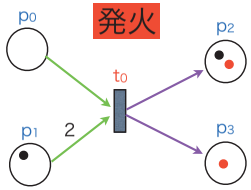
$1 \leqq$ (p_0のトークン数)
&
$2 \leqq$ (p_1のトークン数)
↓ なら
発火可能

Petri Nets

プレースの有限集合 $P:=\{p_0, p_1, \cdots, p_s\}$
トランジションの有限集合 $T:=\{t_0, t_1, \cdots, t_n\}$
Pre, Post : $T \to P \to \{0,1,2,\cdots\}$ アークの重み
$M := P \to \{0,1,2,\cdots\}$　マーキングの型

マーキング　$\overset{p_0\ p_1\ p_2\ p_3}{\langle 0, 1, 2, 1\rangle}$

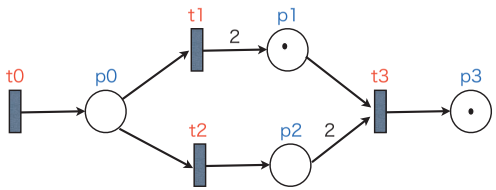
発火



p_0から1個
p_1から2個　トークンを除去

p_2から1個
p_3から1個　トークンを付加

Petri Nets



<0,0,0,0> $\xrightarrow{t0;\ t1;\ t0;\ t2;\ t3}$ <0,1,0,1>

到達可能

Petri Nets

ペトリネットの解析

可達問題

Mi <0,0,0,0> $\xrightarrow{\exists s}$ Md <0,1,0,1>

被覆問題

Mi <0,0,0,0> $\xrightarrow{\exists s}$ <0,2,0,2>

被覆

<0,1,0,1> Md

研究背景

[Karp & Miller, 1969]

Karp-Miller Tree → Minimal Coverability Set

KMTを用いてMCSを求める

計算量が多い

被覆性判定に使える

MCTを構成して効率化 [Finkel, 1991]

アルゴリズムの正当性に誤りを発見し、これを改善 [Luttge, 1995]

改善されたアルゴリズムの停止性に誤りを発見 [Geeraerts, 2007]

最適化で間違いが起き易い

研究背景

Karp-Miller Tree → Minimal Coverability Set

KMTを用いてMCSを求める

KMTのベースとなる部分の形式化

最適化の正当性、停止性の保証

（に貢献できるかも）

今回

KMTの構成におけるKM加速の一部を形式化

Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

ペトリネットの形式化

SSReflect：証明支援系Coqの拡張

有限集合についての性質に強い

有限集合の型が既に用意されている

Def:

`P:finType` プレース

`T:finType` トランジション

ペトリネットの形式化

	関数	tuple
型の定義	`M := P → nat`	`M := #|P|.-tuple nat`
プレースpにおけるトークン数	`m p`	`tnth m (enum_rank p)`
マーキングのequalityと"="との一致	外延性が使えないので一致しない	グラフなので一致する

`m:M p:P`

ペトリネットの形式化

	有限集合上の関数
型の定義	`M := {ffun P → nat}`
プレースpにおけるトークン数	Coercionによって `m p`
マーキングのequalityと"="との一致	実態がグラフなので一致

`m:M p:P`

状態遷移の形式化

マーキングの　比較　計算　がしたい

↕　↕

ffun上の　関係　演算　として定義

状態遷移の形式化

各要素毎の比較

関係のcomponent-wiseへの拡張

cwR = component-wise relation

```
cwR (r:A → B → bool)
    (ma:{ffun P → A}) (mb:{ffun P → B}):=
        [forall (p:P), r (ma p) (mb p)]
```

bool値

有限だから

Notation : `m <=m m' := cwR leq m m'`

状態遷移の形式化

component-wiseな演算

```
ffmap2 (f:A → B → C)
    (ma:{ffun P → A}) (mb:{ffun P → B}) :=
            [ffun (p:P)=> f (ma p) (mb p)]
```

マーキングの加法　　自然数上の加法

↑　　↑

Notation: `m +m m' := ffmap2 addn m m'`

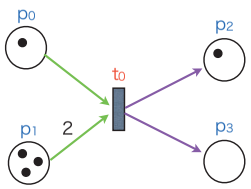
`m -m m' := ffmap2 subn m m'`

↓　　↓

マーキングの減法　　自然数上の減法

状態遷移の形式化

状態遷移



m_0 = <1,3,1,0>

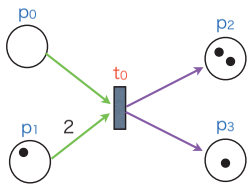
Pre t_0 = <1,2,0,0>

Post t_0 = <0,0,1,1>

発火条件　　Pre t_0 <1,2,0,0> <=m m_0 <1,3,1,0>

状態遷移の形式化

状態遷移



m_0 = <1,3,1,0>

Pre t_0 = <1,2,0,0>

Post t_0 = <0,0,1,1>

発火 m_1 <0,1,2,1> = m_0 <1,3,1,0> -m Pre t_0 <1,2,0,0> +m Post t_0 <0,0,1,1>

状態遷移の形式化

状態遷移

```
trans_one m t
```

Some (m -m (Pre t) +m (Post t))

(Pre t) <=m m

部分関数と考えられる

else → None

発火可能な場合 → 発火後のマーキング

発火不可能な場合 → None

Notation:

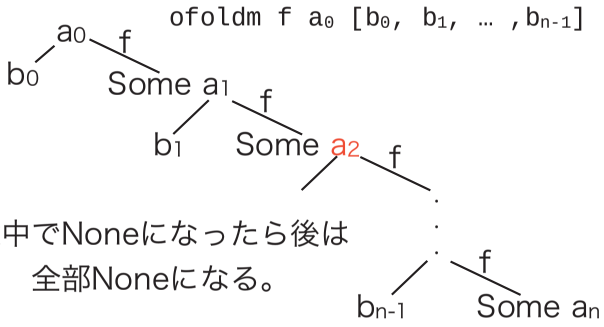
```
m =1{ t }> m' :=(trans_one m t  = Some m')
```

状態遷移の形式化

複数stepの遷移

```
foldM :: (Monad m) => (a -> b -> m a) -> a -> [b] -> ma
foldM f a []     = return a
foldM f a(x:s') = f a x >>= \y -> foldM f y s'
ofoldm {A B:Type} (f:A → B → option A) (a:A)(s:seq B) :=
  if s is x::s' then obind (fun fax => ofoldm f fax s')(f a x) else Some a.
```

ofoldm f a_0 [b_0, b_1, … ,b_{n-1}]



途中でNoneになったら後は
全部Noneになる。

Definition

状態遷移

```
trans_seq := ofoldm trans_one
```

Notation:

```
m ={ s }> m' := trans_seq m s = Some m'
```

```
m ={ [t0,t1,t2,t3] }> m'
```

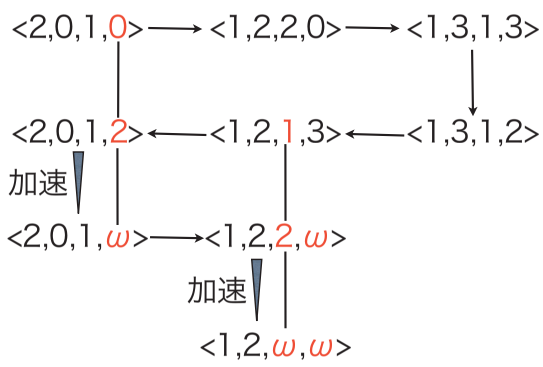
↕

```
m =1{ t0 }> m0 =1{ t1 }> m1 =1{ t2 }> m2
=1{ t3 }> m'
```

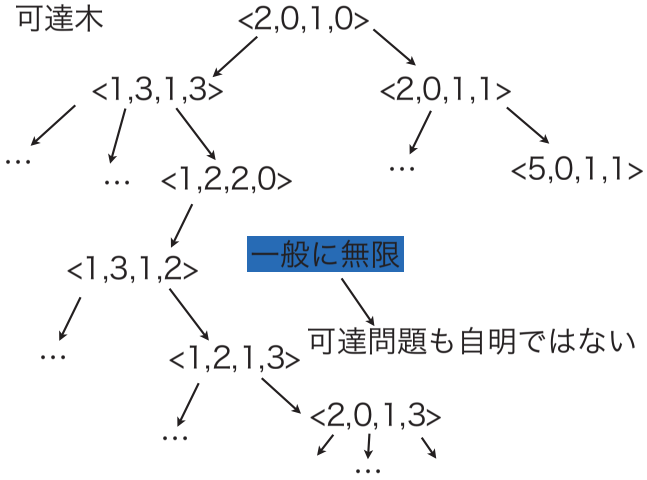
Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

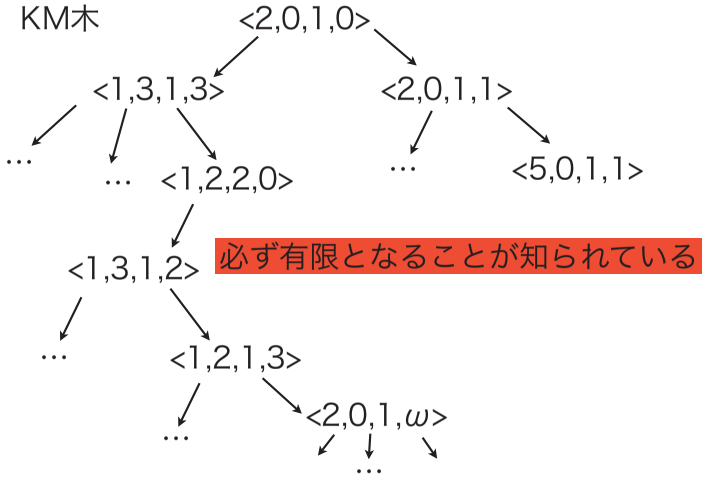
KM加速



KM加速

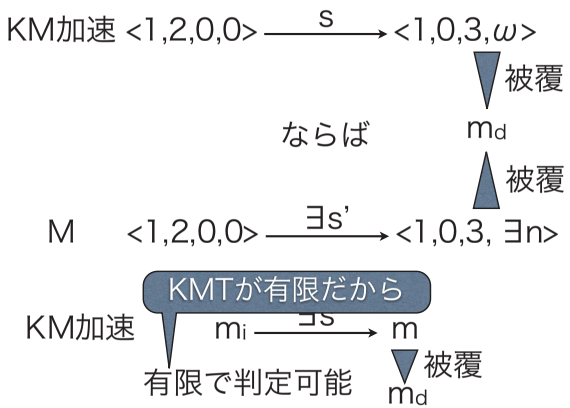


KM加速



KM加速

被覆性判定

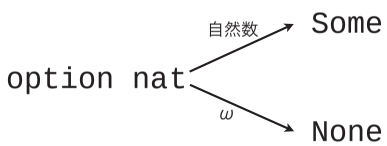


Index

- 研究背景
- 状態遷移系としてのペトリネットの形式化
  - ペトリネットの形式化
  - マーキングの形式化
  - 状態遷移の形式化
- KM加速
- KM加速の形式化
- まとめ

KM加速の形式化

ω付きの自然数の定義

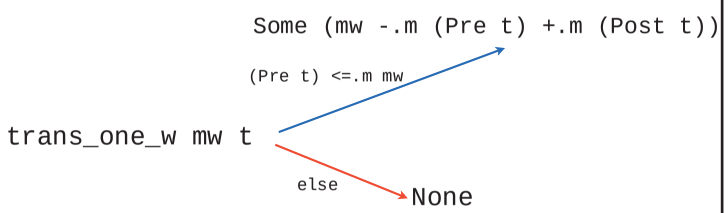


ω付きのマーキングの型

```
Mw:={ffun P -> option nat}
```

KM加速の形式化

ω付きの遷移



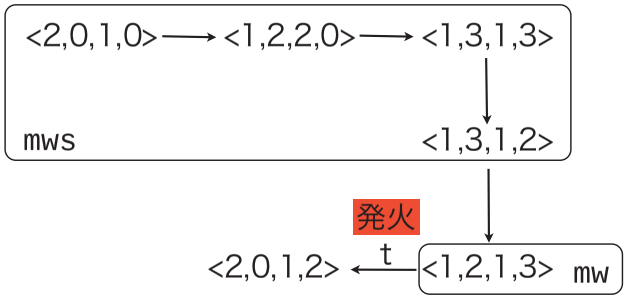

```
trans_seq_w := ofoldm trans_one_w
        m ={w s }> m'
```

KM加速の形式化

履歴付きのマーキングから履歴付きのマーキングを求める

```
KM_step (mws, mw) t:=
  if trans_one_w mw t is Some mw'
          then Some (rcons mws mw,KM_help mw' (rcons mws mw))
          else None.
```

KM_step (mws,mw) t = Some (mws', mw')

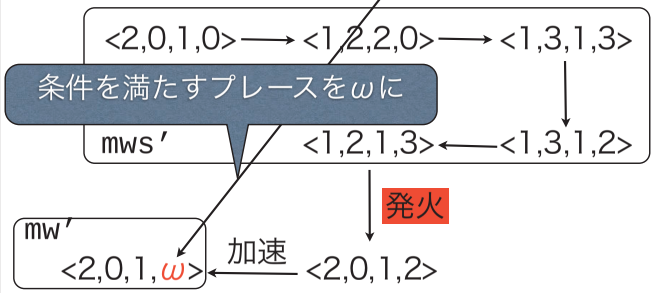


KM加速の形式化

履歴付きのマーキングから履歴付きのマーキングを求める

```
KM_step (mws, mw) t:=
  if trans_one_w mw t is Some mw'
          then Some (rcons mws mw,KM_help mw' (rcons mws mw))
          else None.
```

KM_step (mws,mw) t = Some (mws', mw')



KM加速の形式化

1stepのKM加速ができた

複数stepも

これもoptionを返すので

KM_fun := ofoldm KM_step

まとめ

現在までの成果：ペトリネットの形式化

状態遷移の形式化

KM加速の定義の形式化

被覆性の片方（被覆される方）の形式化

深刻な思い違いが無ければ、後一歩。

今後の課題：被覆性のもう片方

停止性の証明

（これに関してはWQO,Almost-fullが利用できそう[Dim10]）

[Dim10]: Dimitrios et al.,Stop when you are Almost-Full Adventures in constructive termination, 2010