ペトリネットにおける
有界性判定の形式化

稲垣 衛（千葉大学大学院融合理工学府 M1）
山本 光晴（千葉大学大学院理学研究院）

1. ペトリネットについて

2. 被覆性・有界性問題とKarp-Miller木

3. 有界性判定の形式化

4. 今後の課題

1. ペトリネットについて

2. 被覆性・有界性問題とKarp-Miller木

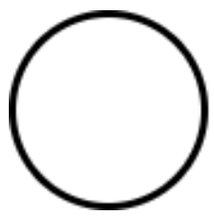
3. 有界性判定の形式化

4. 今後の課題

ペトリネットとは

1962年、ドイツのC.A.Petriにより提唱された、システムのモデル化に使われる有向２部グラフ

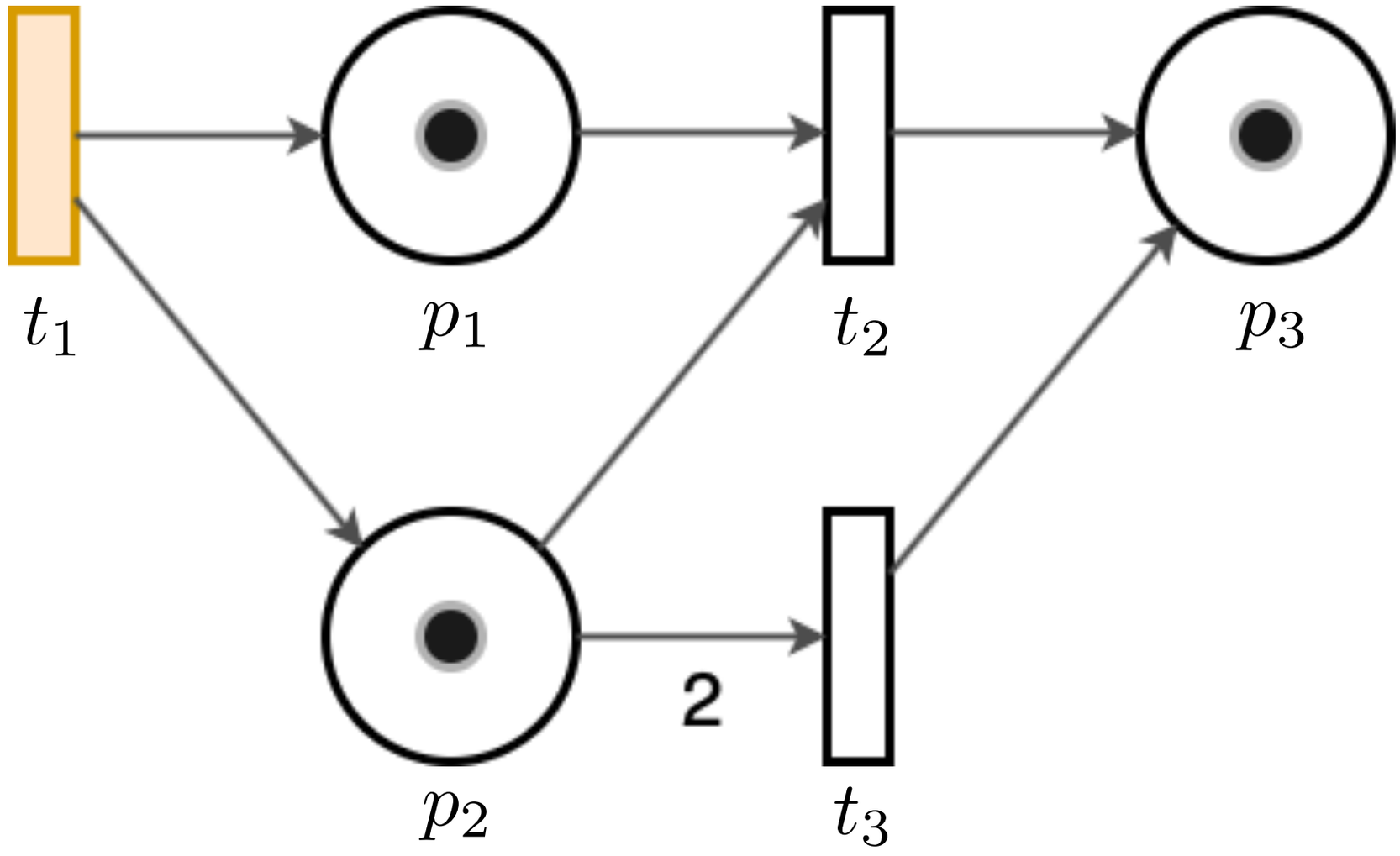
- ノード（頂点）は「プレース」と「トランジション」の2種類が存在する。


：プレース


：トランジション

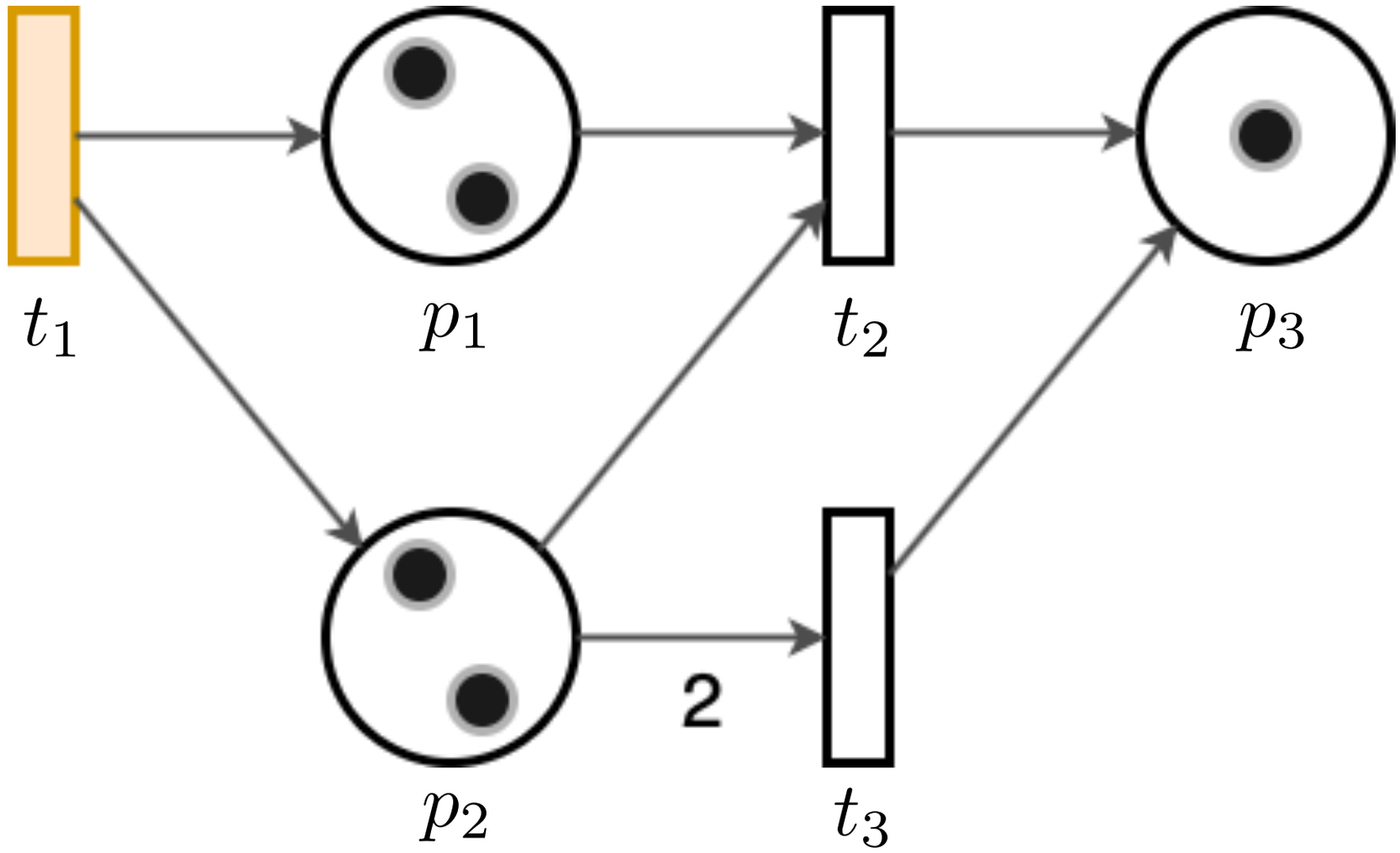
- 並行的、非同期的、非決定的動作の記述に使われる。
- 操作「発火」によって状態「マーキング」が遷移する状態遷移系である。

ペトリネットの具体例



- 各プレースには有限個のトークンが入る
- トランジションが１つ「発火」することでトークン数が変化する。

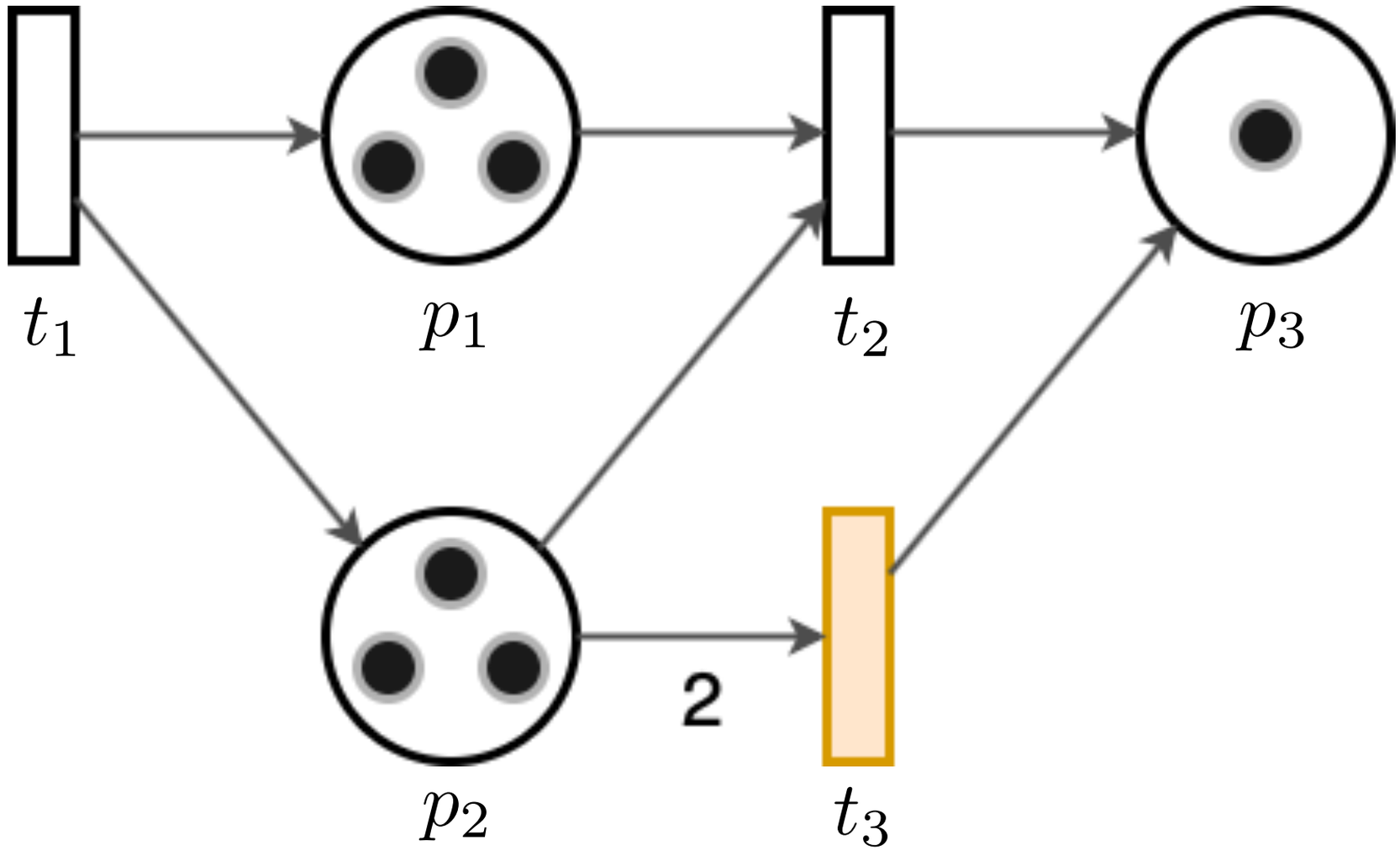
ペトリネットの具体例



$(1,1,1) \xrightarrow{t_1} (2,2,1)$

マーキングとはプレースを受け取り
そのトークン数を返す関数である。

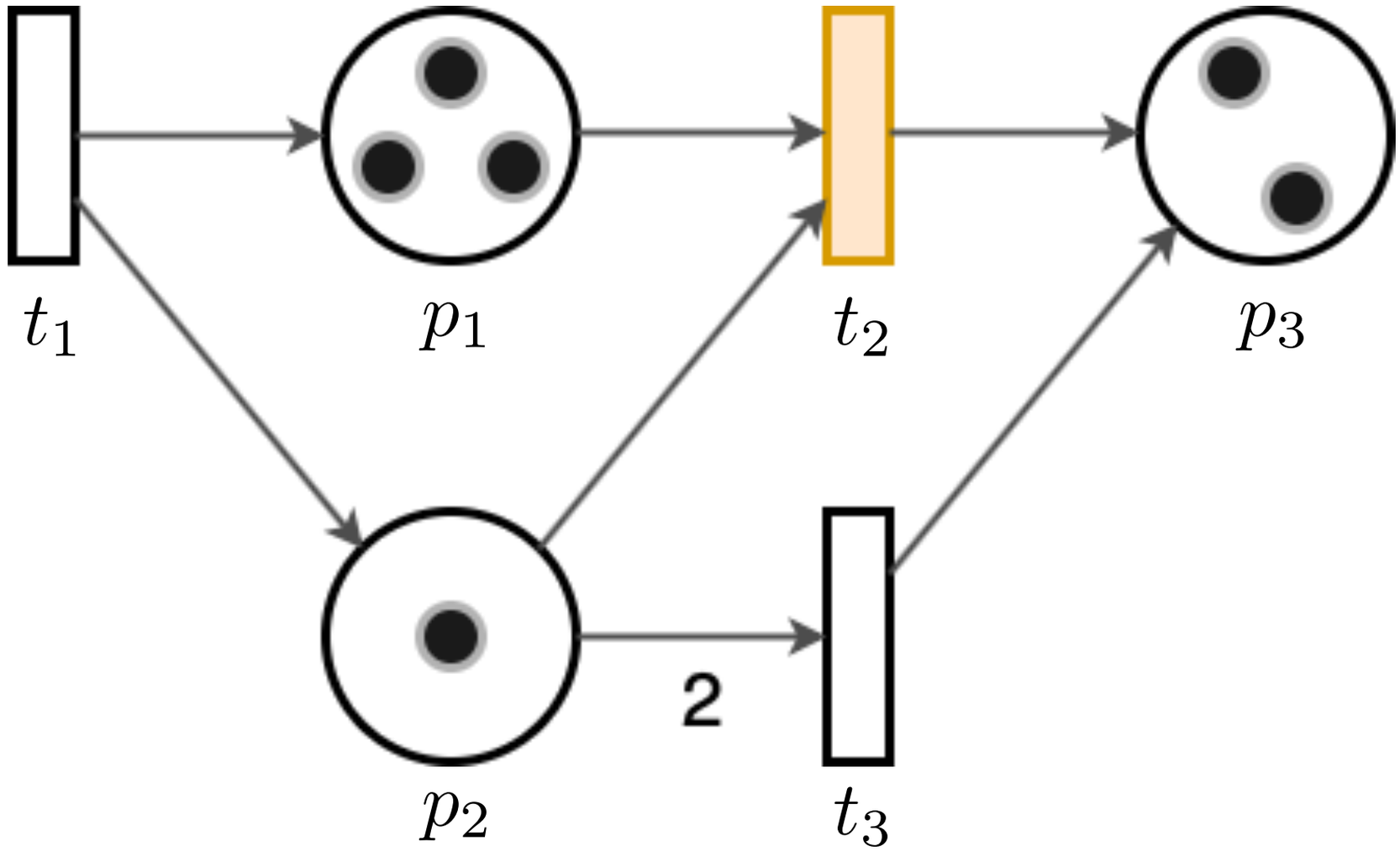
ペトリネットの具体例



$(1,1,1) \xrightarrow{t_1} (2,2,1) \xrightarrow{t_1} (3,3,1)$

辺に数が書いてある場合は
その数だけトークン数が増減する

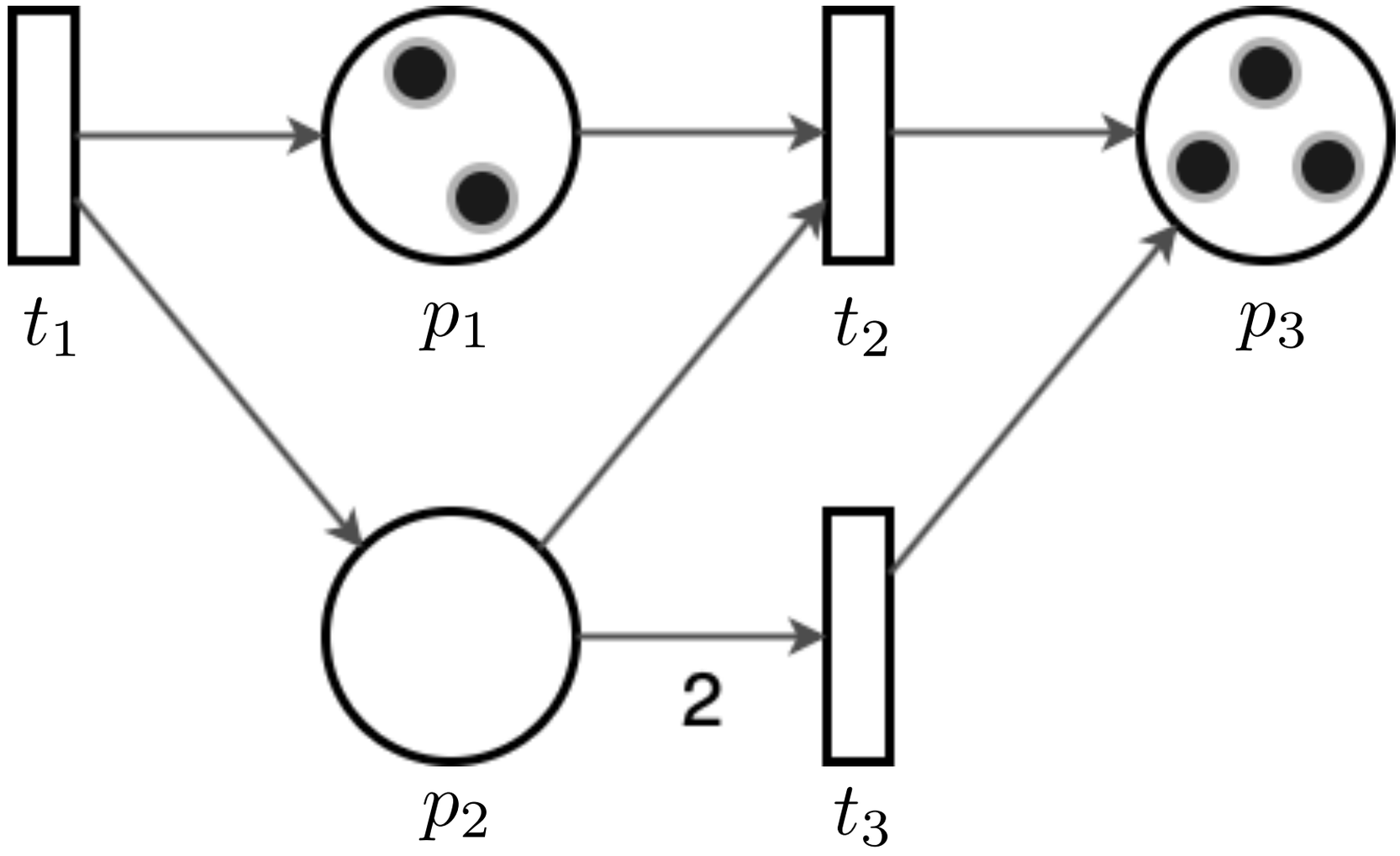
ペトリネットの具体例



$$(1,1,1) \xrightarrow{t_1} (2,2,1) \xrightarrow{t_1} (3,3,1) \xrightarrow{t_3} (3,1,2)$$

この時、p_2 にトークンが
２個以上無いため
t_3 は発火できない。

ペトリネットの具体例



$$(1,1,1) \xrightarrow{t_1} (2,2,1) \xrightarrow{t_1} (3,3,1)$$
$$\xrightarrow{t_3} (3,1,2) \xrightarrow{t_2} (2,0,3)$$

1. ペトリネットについて

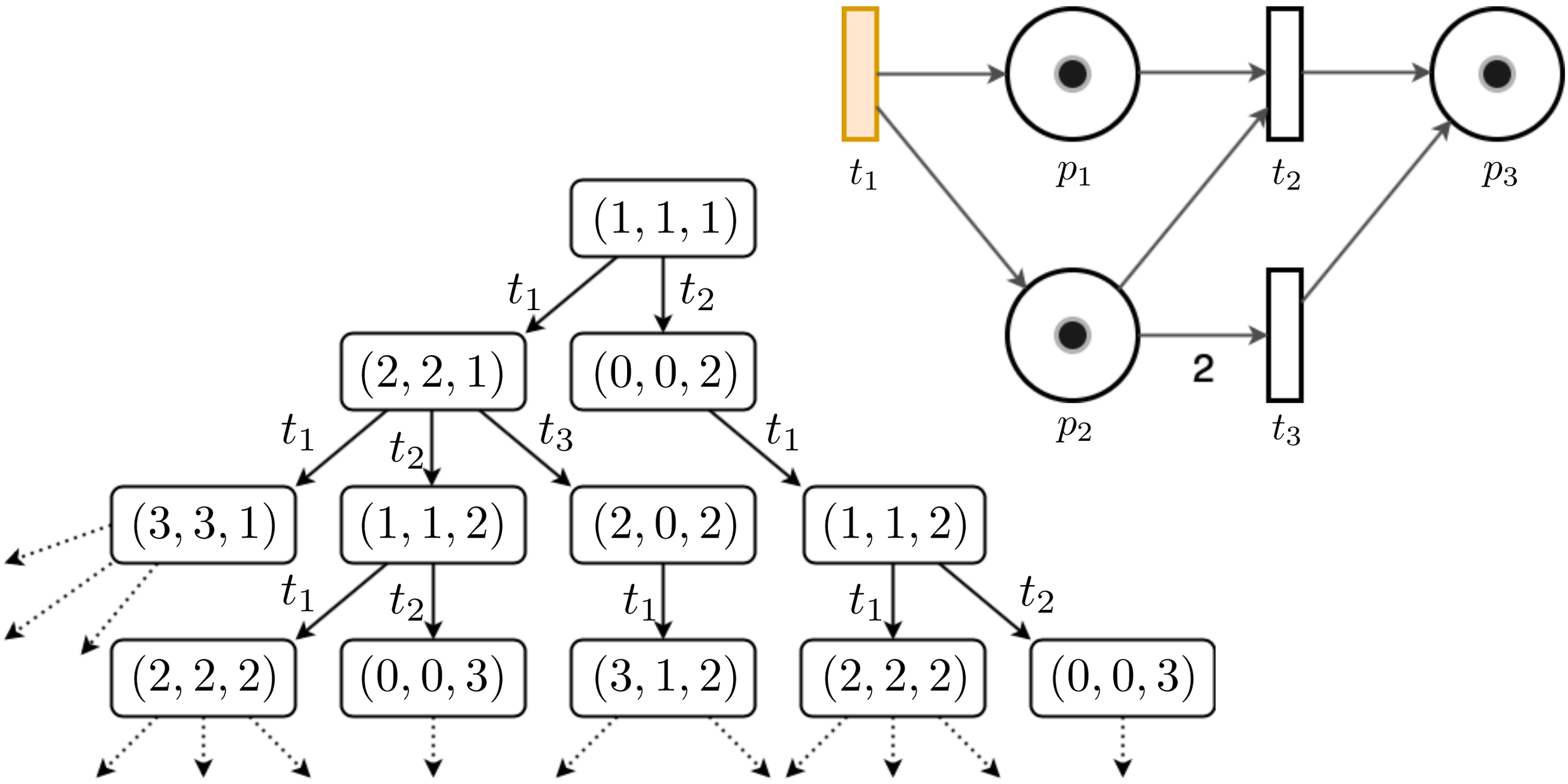
2. 被覆性・有界性問題とKarp-Miller木

3. 有界性判定の形式化

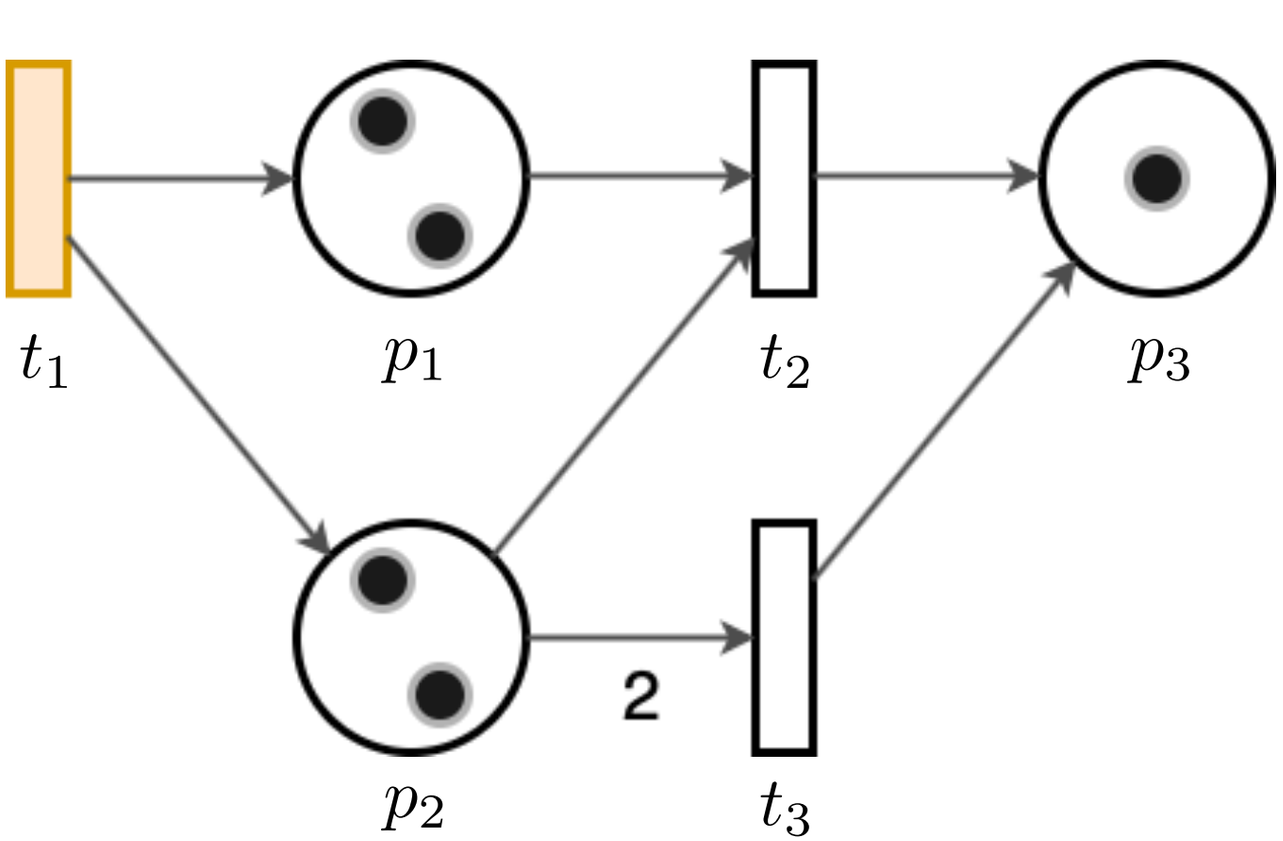
4. 今後の課題

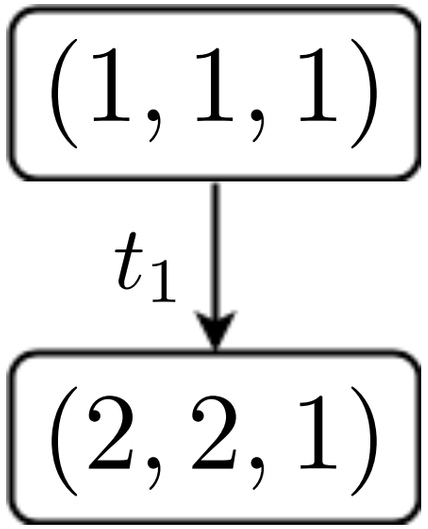
可達木

ペトリネットの状態遷移を木で表したもの



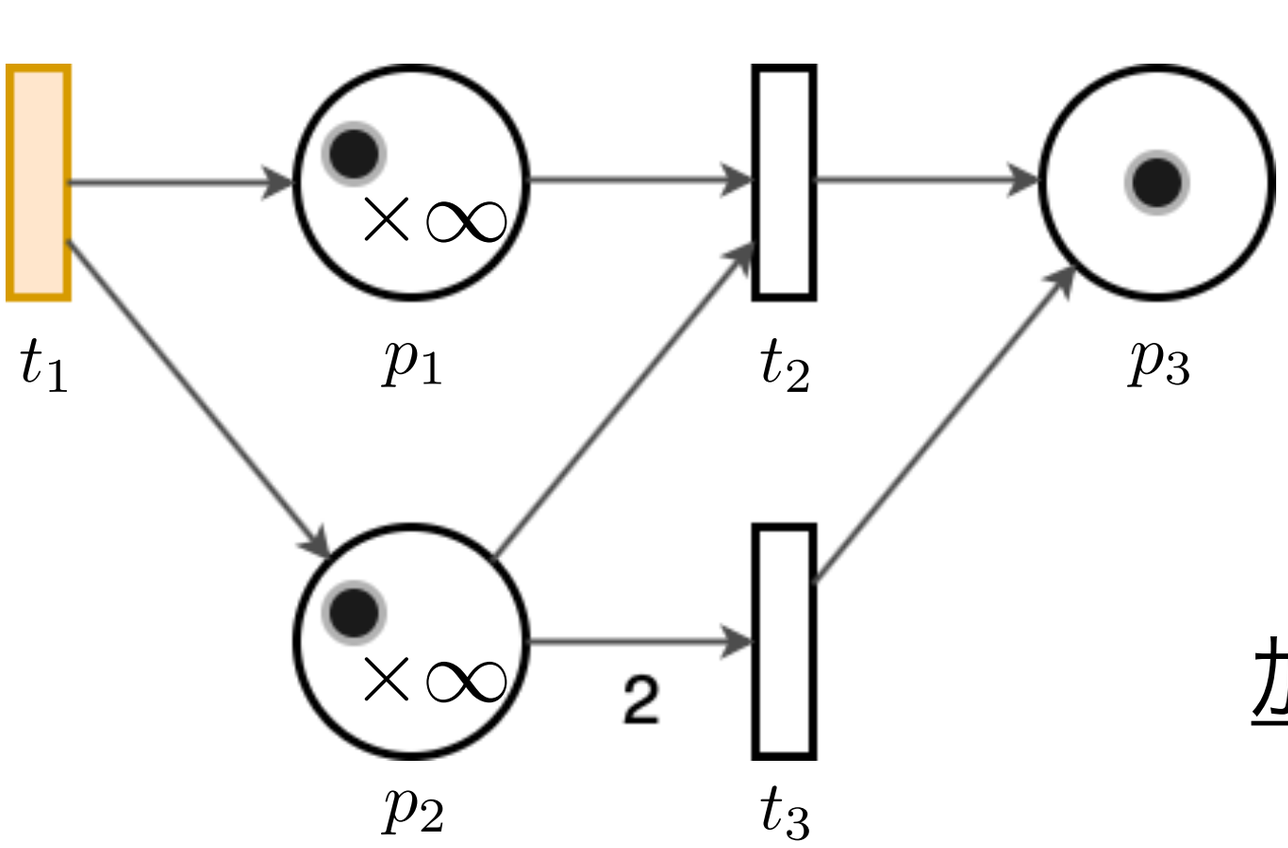
KM加速

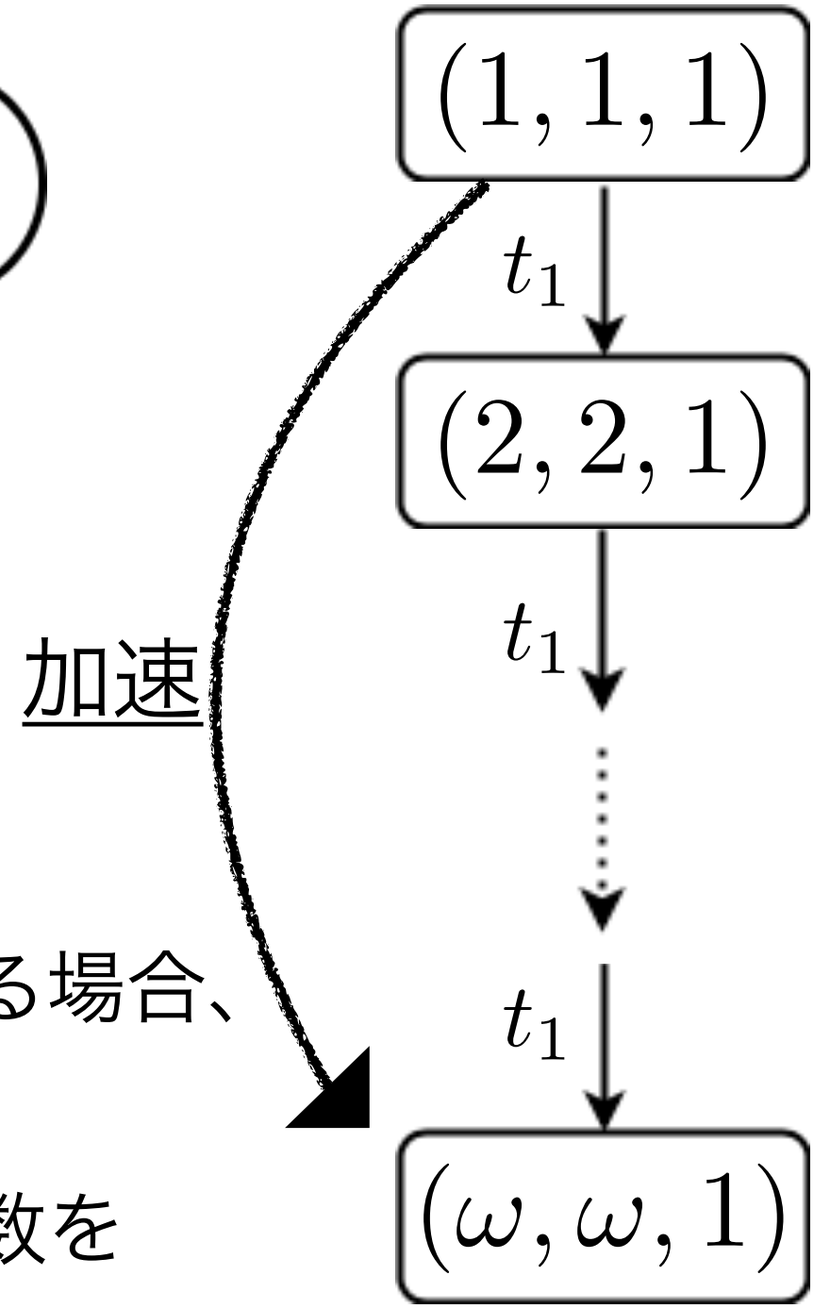




全てのプレースについて、トークン数が同じかそれより多い場合に、「被覆する」という。 $(1,1,1) \leq (2,2,1)$

KM加速

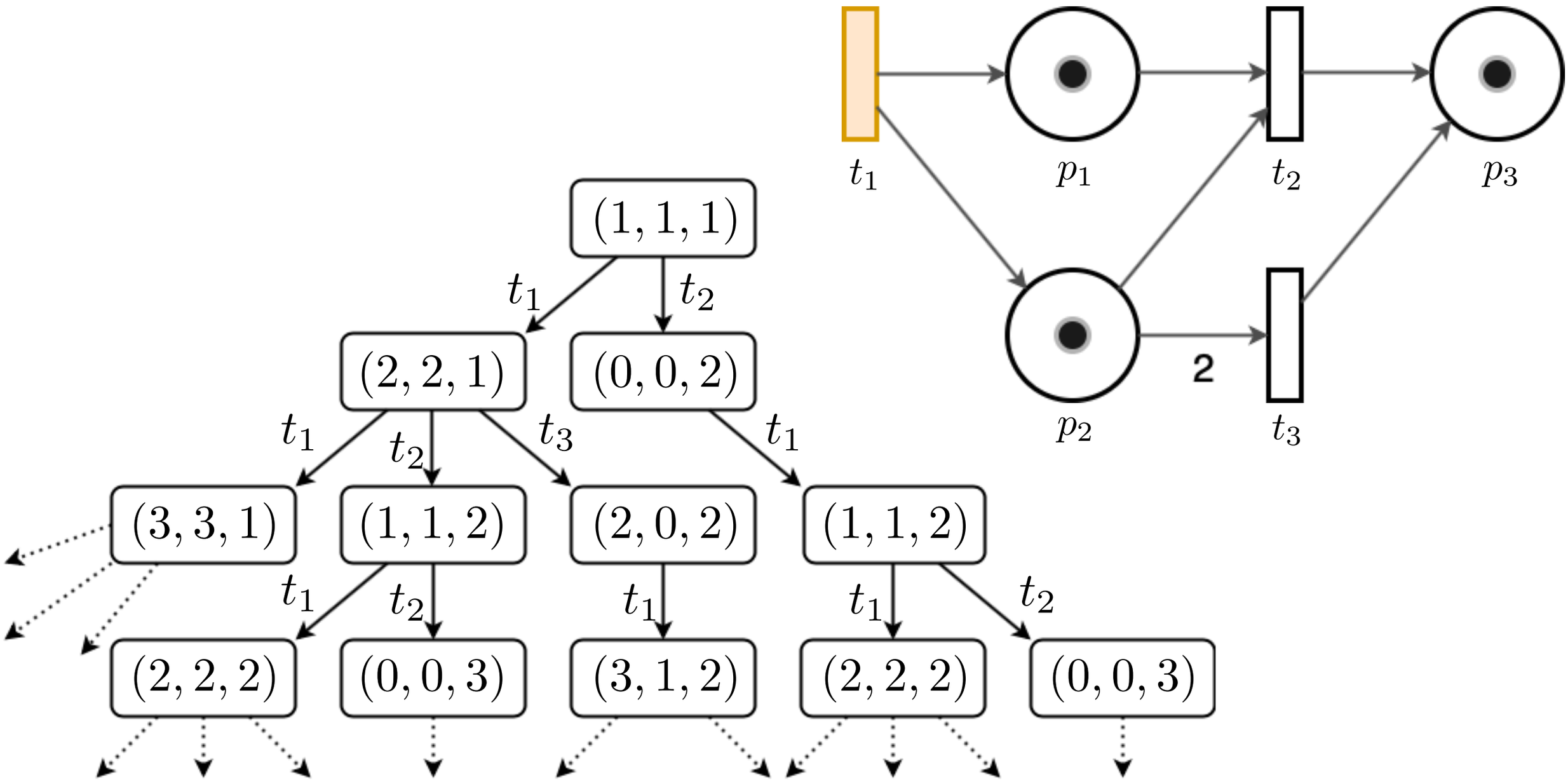




被覆されるマーキングが先祖に存在する場合、
そこからの発火系列を繰り返すことで、
真に大きくなっている部分のトークン数を
限りなく増やすことができる。

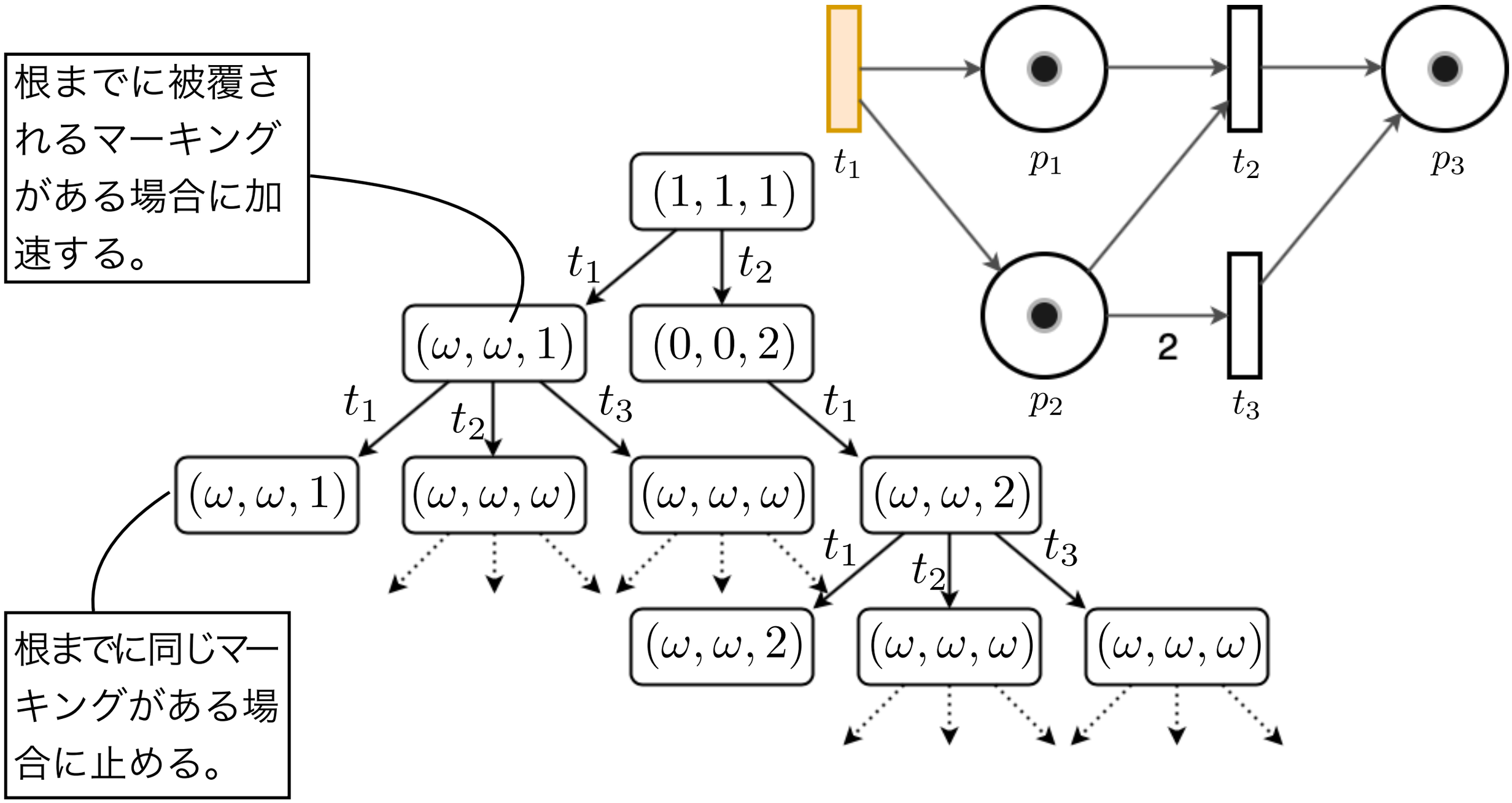
可達木

ペトリネットの状態遷移を木で表したもの



Karp-Miller木(1967)

KM加速を用いたω付きの状態遷移を木で表したもの

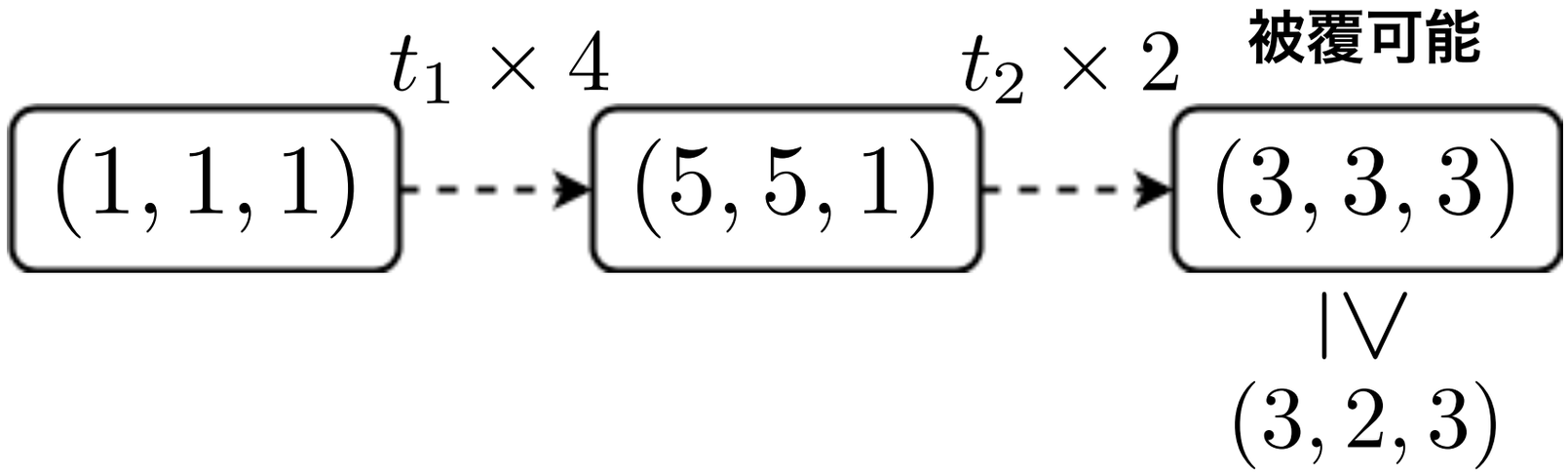


被覆性問題とKM木

ペトリネットと初期マーキングが与えられている。

被覆性問題　　あるマーキング m を定め、
それを被覆するマーキングに到達できるか。

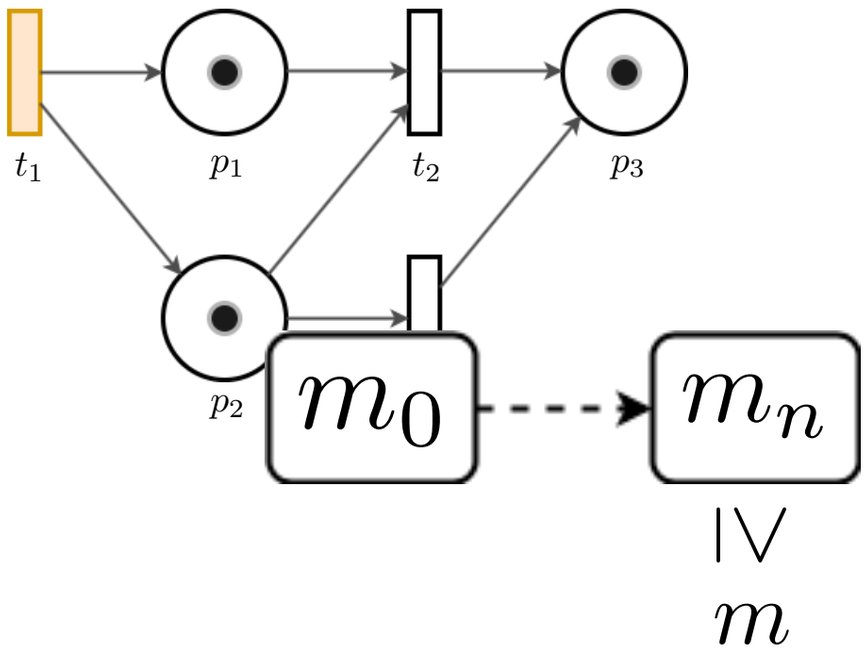
例　先ほどのペトリネットと初期マーキングにおいて
$(3, 2, 3)$ を被覆するマーキングに到達できるか



被覆性問題とKM木

被覆性問題はKM木によって決定可能である。

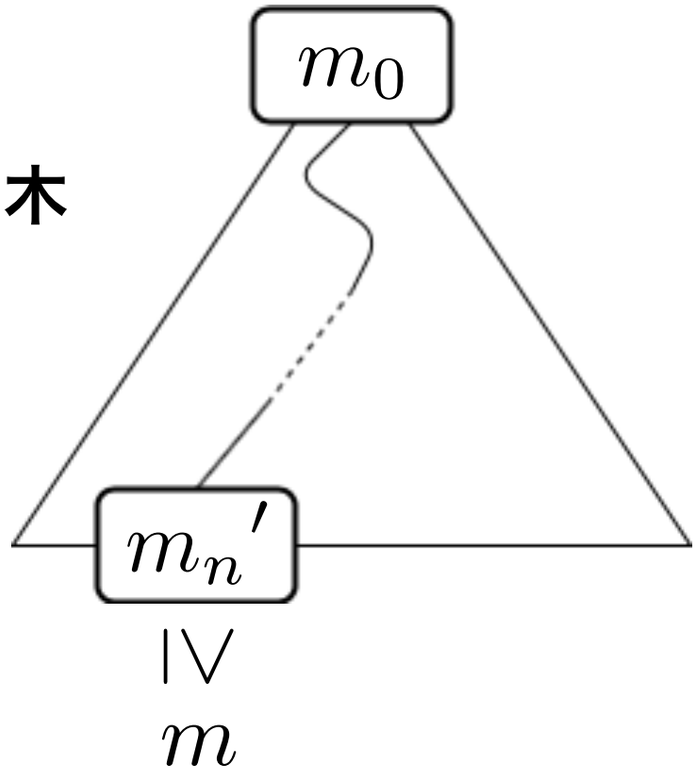
ペトリネット



完全性

健全性

KM木

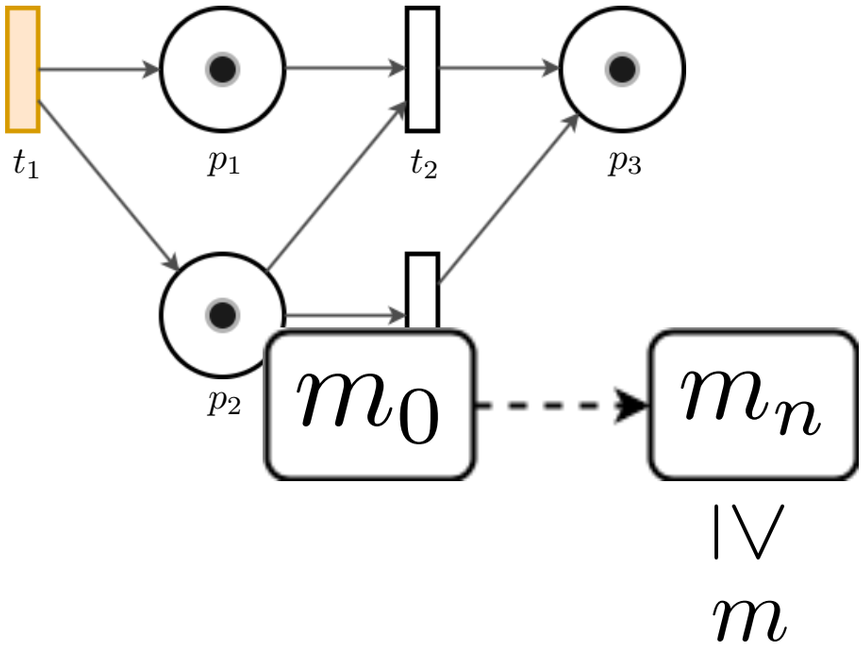


ペトリネットにおいて m を被覆するマーキングに
到達可能であるならば、KM木において
m を被覆するマーキングが存在する。

被覆性問題とKM木

被覆性問題はKM木によって決定可能である。

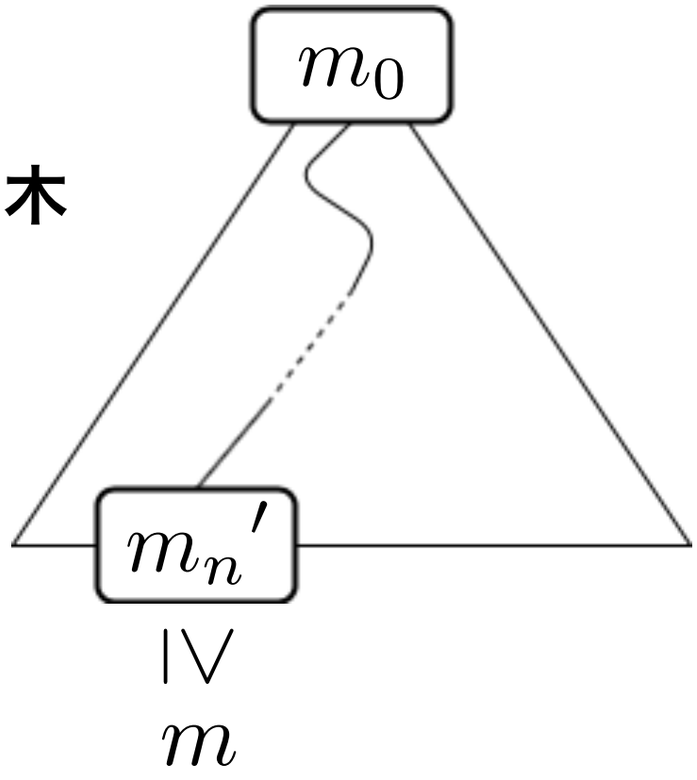
ペトリネット



完全性

健全性

KM木



KM木に mを被覆するマーキングが存在するならば、
ペトリネットにおいても m を被覆するマーキングに
到達可能である。

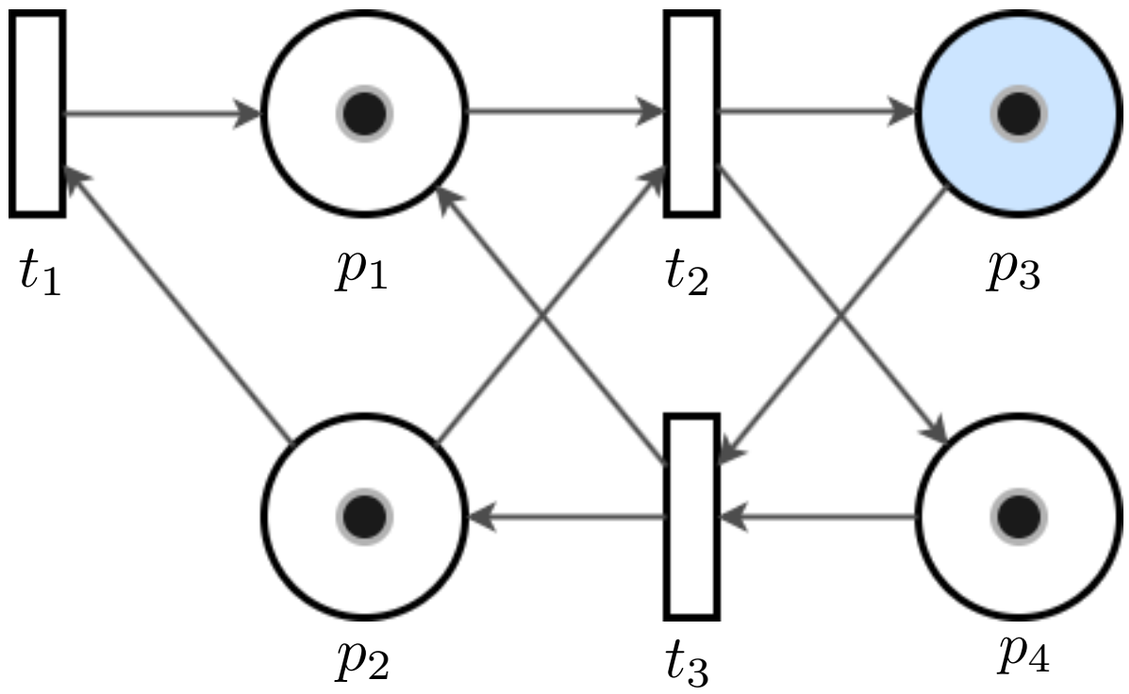
有界性問題とKM木

ペトリネットと初期マーキングが与えられている。

（プレース）有界性問題

指定されたプレースにおいて、トークン数の上限が存在するか。

例



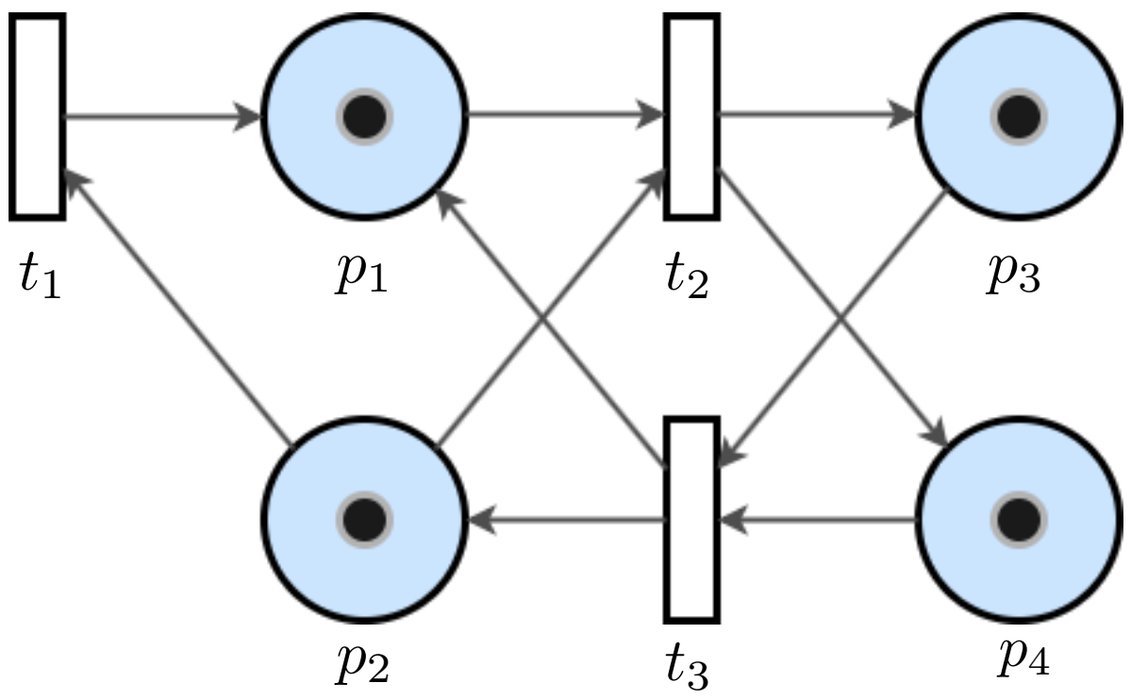
有界性問題とKM木

ペトリネットと初期マーキングが与えられている。

有界性問題

任意のプレースにおいて、トークン数の上限が存在するか。

例



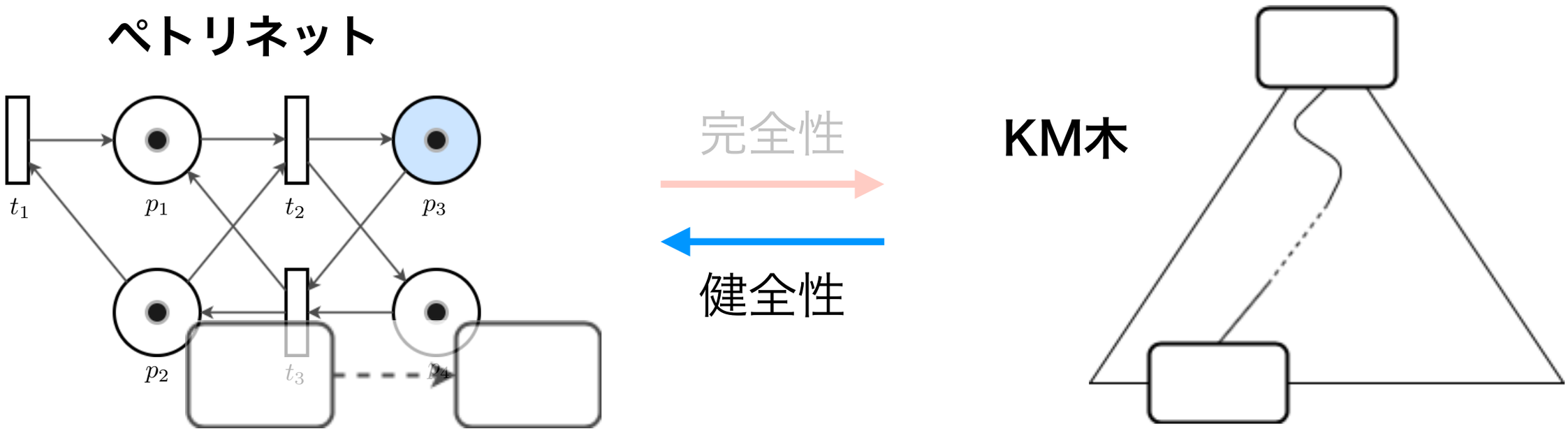
有界性問題とKM木

（プレース）有界性問題はKM木によって決定可能である。

ペトリネットにおいて指定したプレースに上限が存在するならば、
KM木でそのプレースがωになるマーキングが存在しない。

有界性問題とKM木

（プレース）有界性問題はKM木によって決定可能である。



KM木でそのプレースがωになるマーキングが存在しないならば、
ペトリネットにおいて指定したプレースに上限が存在する。

有界性問題とKM木

有界性問題はKM木によって決定可能である。

ペトリネットにおいて任意のプレースに上限が存在するならば、
KM木で任意のプレースにおいてωとなるマーキングが
存在しない。

有界性問題とKM木

有界性問題はKM木によって決定可能である。

KM木で任意のプレースにおいてωとなるマーキングが
存在しないならば、ペトリネットにおいて任意のプレースに
上限が存在する。

1. ペトリネットについて

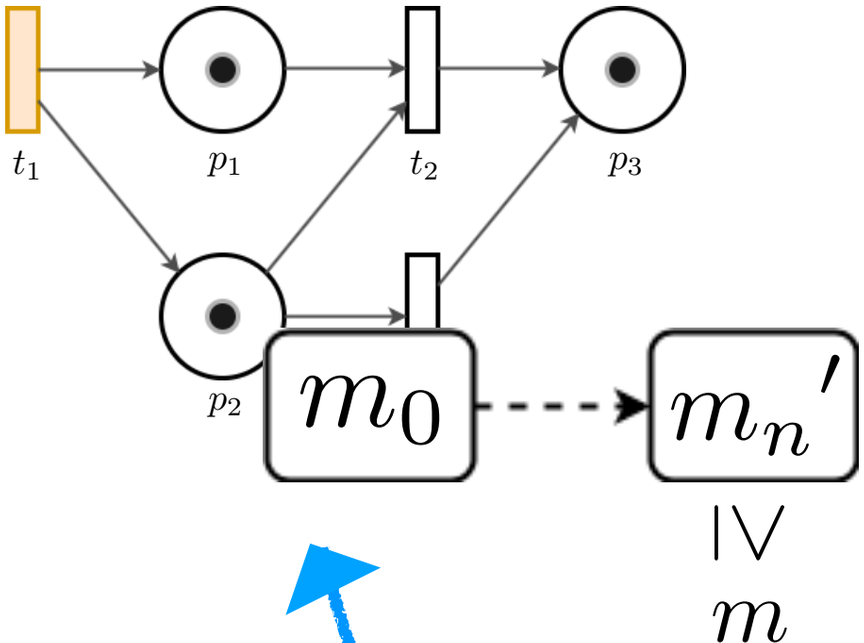
2. 被覆性・有界性問題とKarp-Miller木

3. 有界性判定の形式化

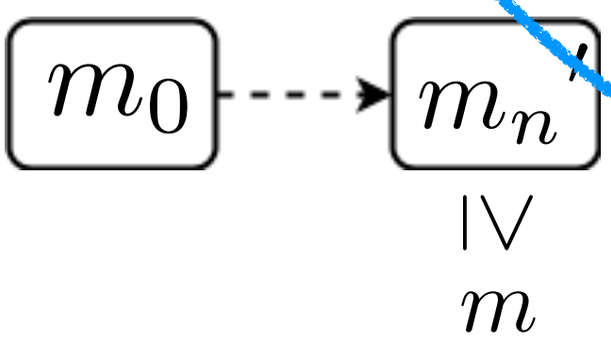
4. 今後の課題

被覆性における既存研究

ペトリネットの状態遷移系

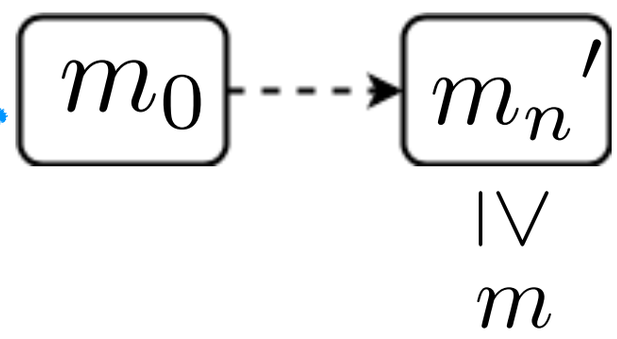


ω付き・加速なし遷移系



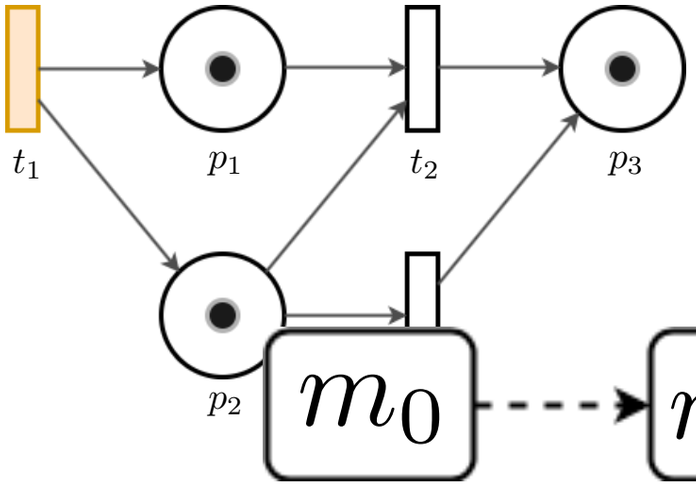
TPP2014 関根翔吾
Coq/SSReflectによる
健全性の主要部分の形式化

ω付き・加速あり遷移系



被覆性における既存研究

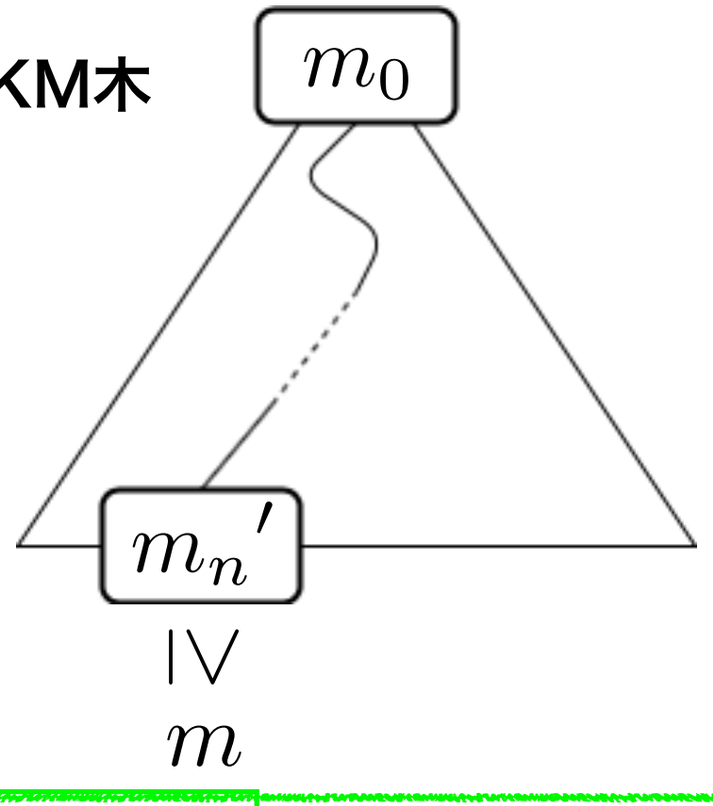
ペトリネットの状態遷移系



完全性

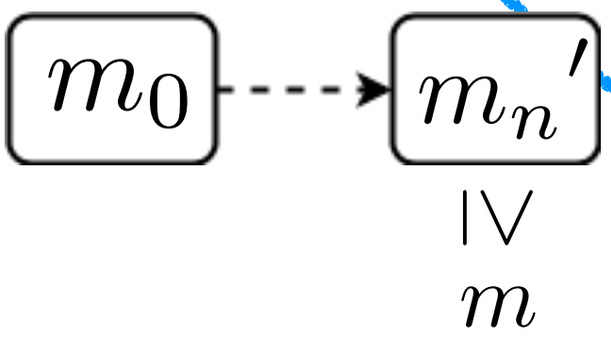
健全性

KM木

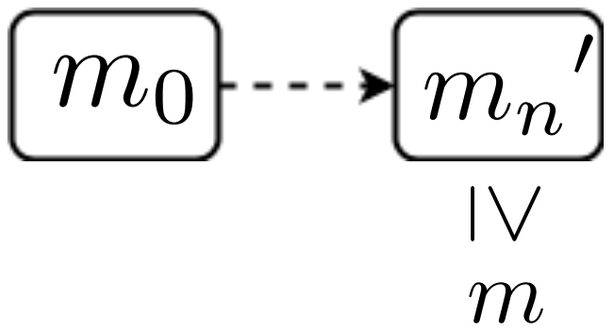


TPP2015 松本早貴
KM木の構成関数

ω付き・加速なし遷移系

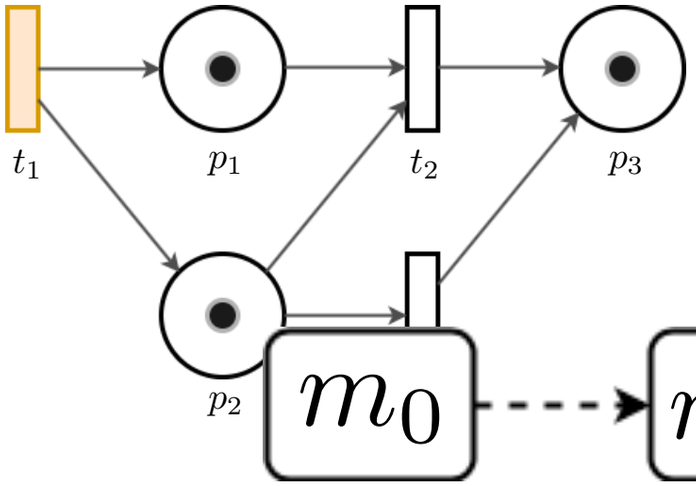


ω付き・加速あり遷移系



被覆性における既存研究

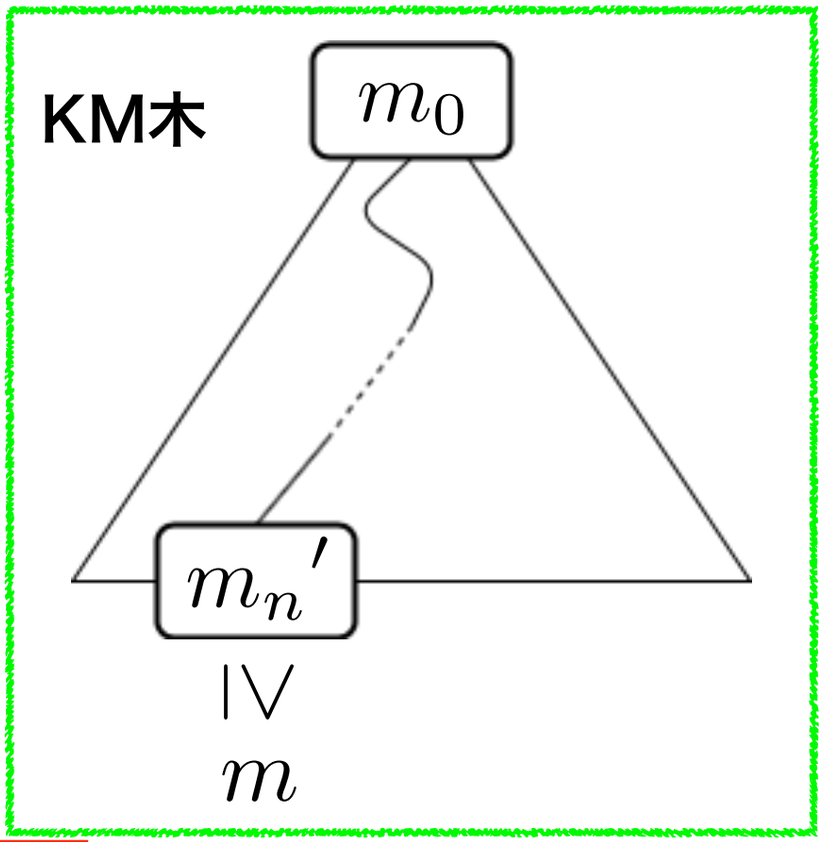
ペトリネットの状態遷移系



完全性

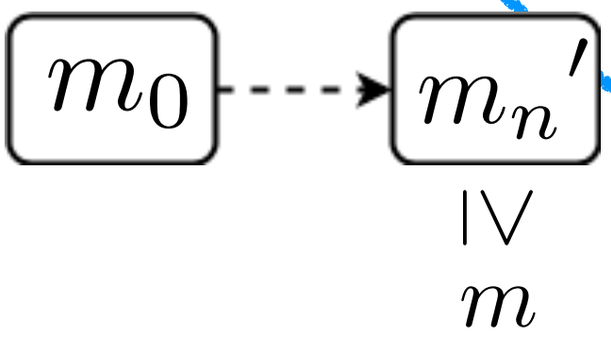
健全性

KM木

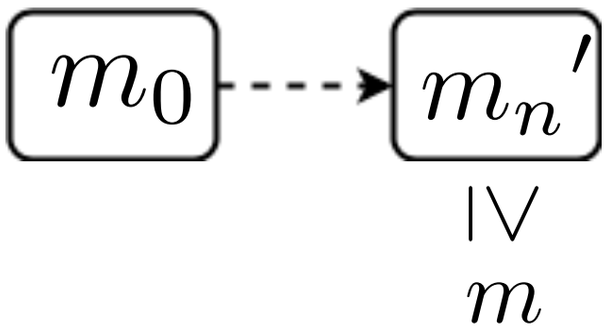


TPP2016 松本早貴
被覆性判定の正当性の形式化

ω付き・加速なし遷移系



ω付き・加速あり遷移系



プレース有界性の定義の方法

SSReflectでは、”reflect （命題）（命題の決定手続き）”と書くことで決定手続きを形式化することができる。

ペトリネットはsection variableによって固定されているため、補題の引数としては受け取っていない。

```
Theorem mkkmtree_boundedP m0 p :
  reflect {for p, bounded m0}
          (~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                         (mkkmtree (mkkmt_init m0))).
```

プレース有界性の定義の方法

```
Definition bounded m0 :=
  forall p, exists n, forall m, reachable m0 m -> m p <= n
```

SSReflectにおける{for y, P1} ⇔ Qx{y / x}を利用して、
プレース **p** における（プレース）有界性は**{for p, bounded m0}**
有界性は**bounded m0**と書くことができる。

```
Theorem mkkmtree_boundedP m0 p :
  reflect {for p, bounded m0}
          (~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                         (mkkmtree (mkkmt_init m0))).
```

プレース有界性の定義の方法

kmtree_hasは、ω付きマーキングに対する述語**P**と、
KM木**tree**を受け取り、Pを満たすマーキングが**tree**の中に
存在するかを判定する手続きである。

markingcはω付きのマーキングの型である。

ここで**natc**とはnat（自然数型）にωを付け加えて
option型にしたものである。(**top**はωの別名）

```
Theorem mkkmtree_boundedP m0 p :
  reflect {for p, bounded m0}
          (~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                         (mkkmtree (mkkmt_init m0))).
```

プレース有界性の定義の方法

(mkkmtree (mkkmt_init m0))は初期マーキング**m0**から
KM木を構成している。

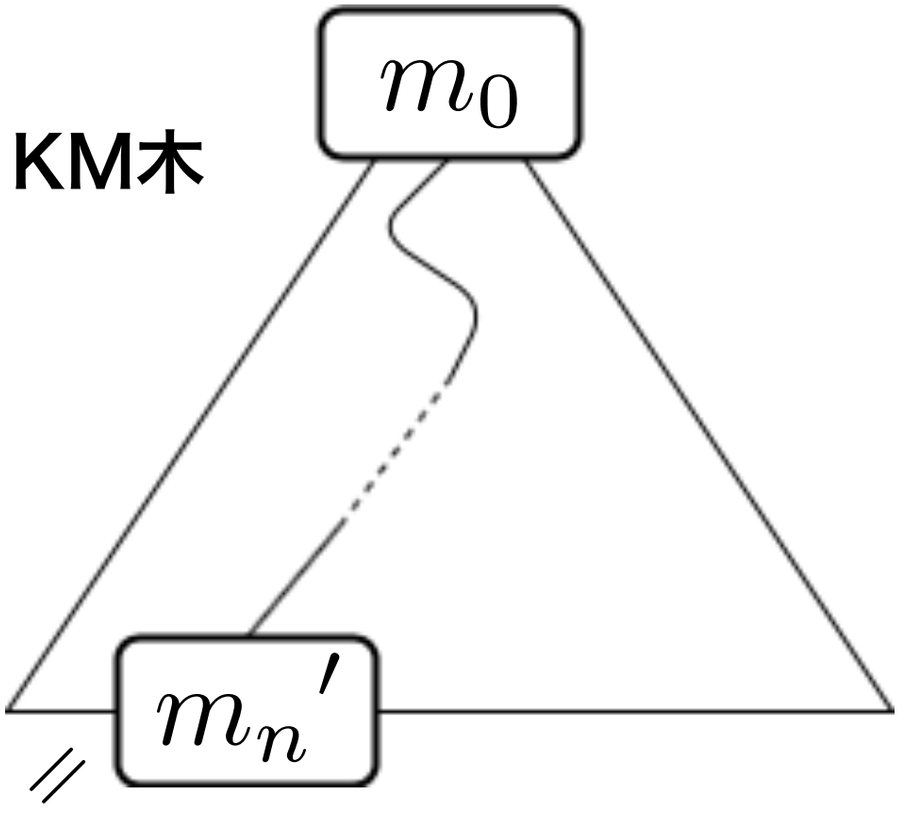
以上で、KM木がプレース **p** でωになるマーキングを
持たないことを判定する手続きを書くことができる。

```
Theorem mkkmtree_boundedP m0 p :
  reflect {for p, bounded m0}
          (~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                         (mkkmtree (mkkmt_init m0))).
```

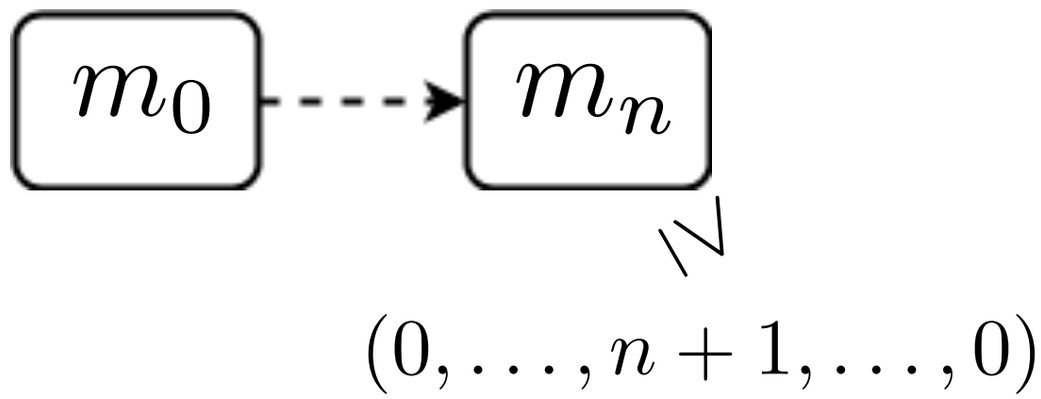
証明の方針（完全性）

ペトリネットのプレース p において、
上限 n が存在するならば、
KM木の中でプレース p でωになるマーキングは
存在しない。（完全性）

KM木



ペトリネットの状態遷移系

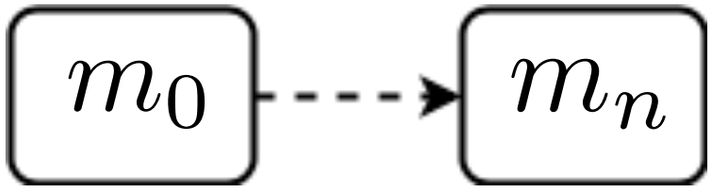


被覆性判定の健全性を
利用する

証明の方針（健全性）

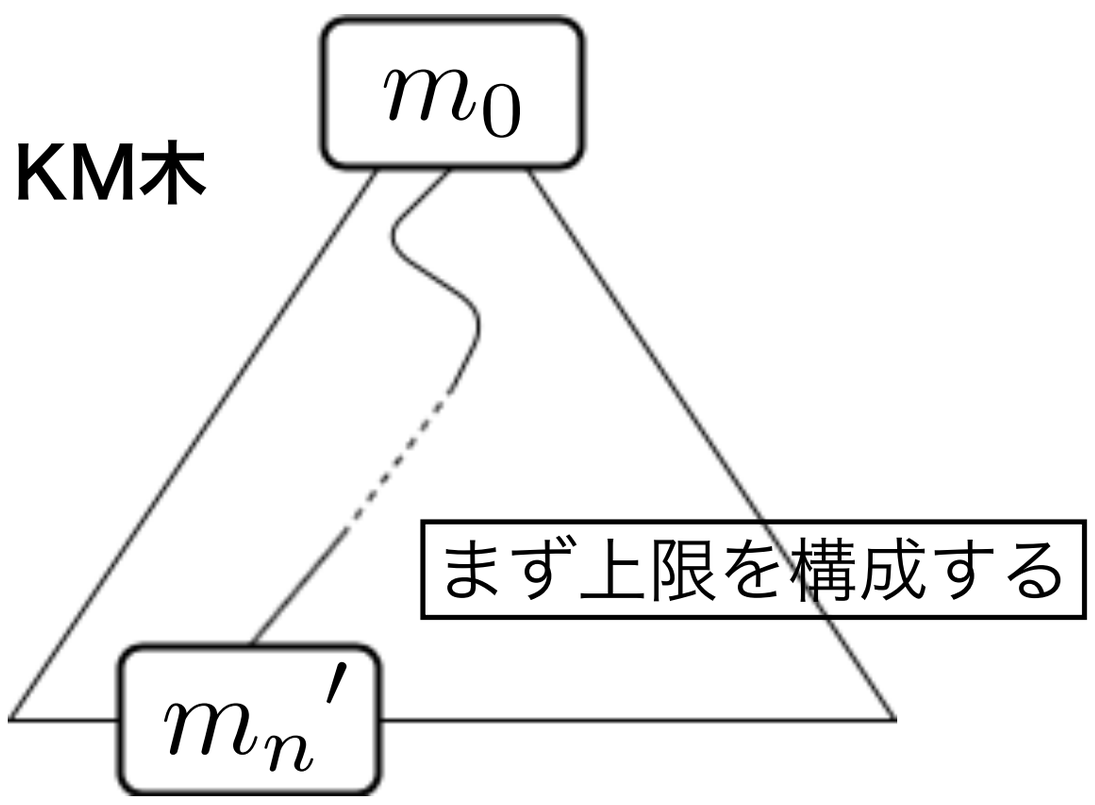
KM木の中でプレース p でωになるマーキングは
存在しないならば、
ペトリネットのプレース p において、
上限が存在する。（健全性）

ペトリネットの状態遷移系



被覆性判定の完全性を
利用する

KM木



証明の方針（健全性）

使用する関数・補題について

上限を構成する関数

```
kmtree_maximum (p : place) (tree : kmtree) :=
\join_(mc <- kmtree_flatten tree) mc p.
```

ここで、**mc p**はトークン数を表すただの自然数ではなく、
ωを含むnatc型である。そのため、
自然数における最大値をとる\maxは使えない。

そのため、順序関係一般のライブラリであるorder.v内にある
束における結びを表す**\join**を利用して、上限を構成する。

order.v (https://github.com/math-comp/finmap/blob/master/order.v)

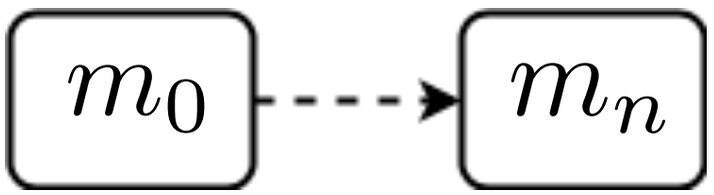
証明の方針（健全性）

使用する関数・補題について

KM木の中のマーキングは上限を超えない

```
Lemma kmtree_maximum_coverable (p : place) (tree : kmtree) mc :
  mc \in tree -> mc p <= kmtree_maximum p tree.
```

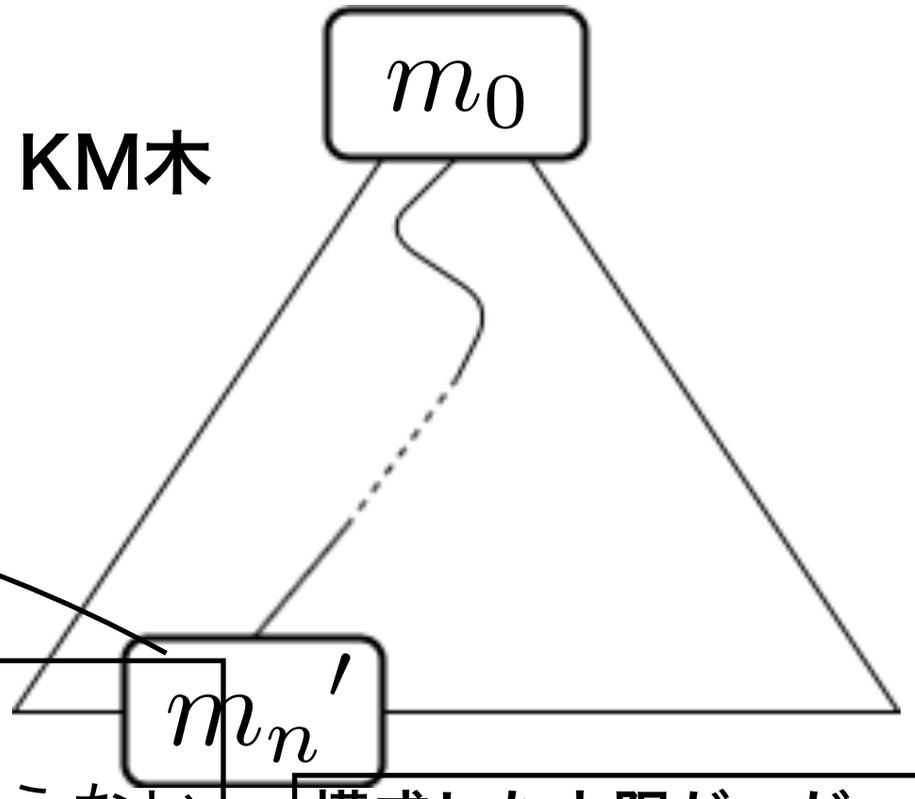
ペトリネットの状態遷移系



m_nを被覆する
マーキングが存在

そのマーキングは
KM木で構成した上限を超えない

KM木



構成した上限がωだったら？

証明の方針（健全性）

使用する関数・補題について

KM木の中に必ず最大値が存在する

```
Lemma kmtree_has_maximum (p : place) (tree : kmtree) :
                (exists mc, mc \in tree) ->
kmtree_has (fun mc : markingc => mc p == (kmtree_maximum p tree)) tree.
```

この補題の仮定はKM木が少なくとも１つは
マーキングを持つことを保証するためのものである。

構成した上限がωならば、KM木の中にプレース ｐ でωになる
マーキングが存在することを導く。

証明の方針（健全性）

使用する関数・補題について

KM木の中に必ず最大値が存在する

```
Lemma kmtree_has_maximum (p : place) (tree : kmtree) :
                (exists mc, mc \in tree) ->
kmtree_has (fun mc : markingc => mc p == (kmtree_maximum p tree)) tree.
```

```
Lemma eq_bigmax (I : finType) F : #|I| > 0 -> {i0 : I | \max_i F i = F i0}.
```

natであれば、補題eq_bigmaxを使うことができるが、
natcでは全順序であることを利用して証明する。

プレース有界性から有界性へ

（プレース）有界性判定の形式化

```
Theorem mkkmtree_boundedP m0 p :
  reflect {for p, bounded m0}
            (~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                           (mkkmtree (mkkmt_init m0))).
```

有界性判定の形式化

```
Theorem mkkmtree_all_place_boundedP m0 :
  reflect (bounded m0)
            [forall p,  ~~ kmtree_has (fun mc : markingc => mc p == top :> natc)
                                      (mkkmtree (mkkmt_init m0))].
```

‘forall_*view*について

```
Variables (T : finType) (P : pred T) (PP : T -> Prop).
Hypothesis viewP : forall x, reflect (PP x) (P x).

Lemma forallPP : reflect (forall x, PP x) [forall x, P x].

Notation "'forall_ view" := (forallPP (fun _ => view)).
```

‘forall_*view*を使うことで、すでにreflectで形式化した命題の
forallで全称量化したものを使うことができる。その結果、
有界性の証明は1行で完了する。

```
Proof.
  by apply: (iffP 'forall_mkkmtree_boundedP).
Qed.
```

1. ペトリネットについて

2. 被覆性・有界性問題とKarp-Miller木

3. 有界性判定の形式化

4. 今後の課題

今後の課題

- 有界であることと、到達可能なマーキングが有限集合であることが同値になることの形式化

- natcにおけるeq_bigmaxが、全順序であるならば一般的に言える性質であることをいう。

- very-WSTS[Blondin+ 17]におけるideal Karp-Miller アルゴリズムと、それによる被覆性及び反復被覆性の判定の形式化