時相論理式を用いた抽象化法のツール化に向けて

田辺 良則† 高橋 孝一‡ 高井 利憲†

† 科学技術振興機構，CREST
‡ 産業技術総合研究所

2004年2月5日
シンポジウム「システム検証の科学技術」

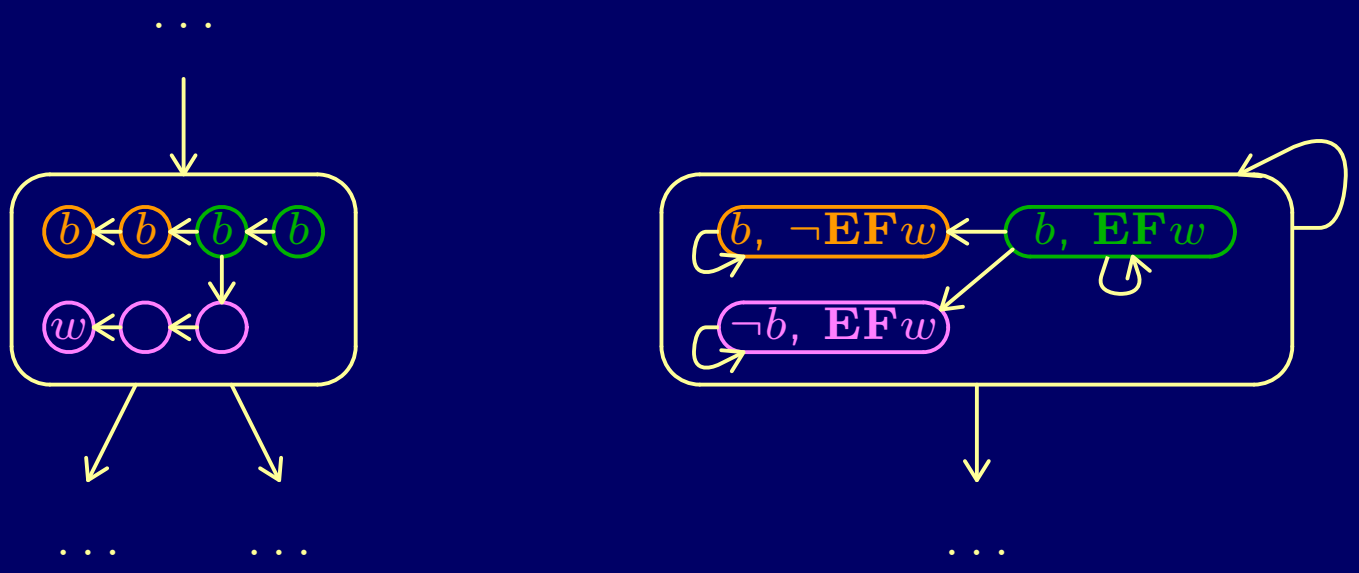
述語抽象化ツール

- 実現例が出つつある．
- 数値型に関する性質の検証には有用．
  （例）
  - 変数 x の値は負にはならない．
  - ラベル L には到達しない．
  - 関数 func1 は関数 func2 より前に呼ばれる．
- ポインタによるデータ構造に関する性質は扱えない．
  （例）
  - いつでも，ポインタをたどるループは作られない．
  - ゴミはいつかは回収される．

時相論理式を用いた抽象化

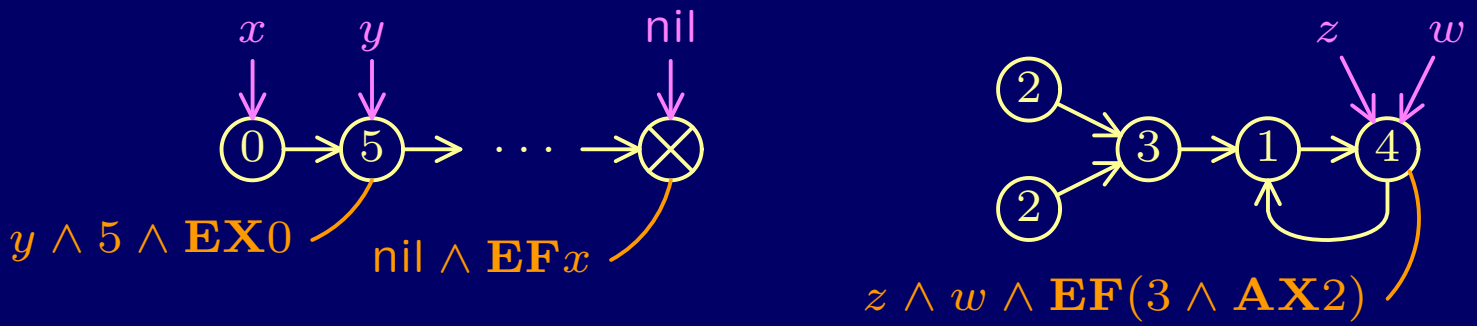
- グラフ書替に関する抽象化技法
- グラフをクリプケ構造と見て，セルの性質を時相論理式で記述:「抽象セル」
- 抽象モデルでの「状態」は，とりうる抽象セルの集合．

具体モデル $\xrightarrow[\{\text{“}b\text{”},\ \text{“}\mathbf{EF}w\text{”}\}]{}$ 抽象モデル



ツール化のための設定

- 各セルは 1 つのポインタと 1 つの値 (整数) を持つ.
- セルを指す変数がある.
- 論理式は CTL.
  - 原子論理式は,「値がいくつ」と「変数に指される」.
  - クリプケ構造の意味での「遷移」は，ポインタの向きと反対に取る.



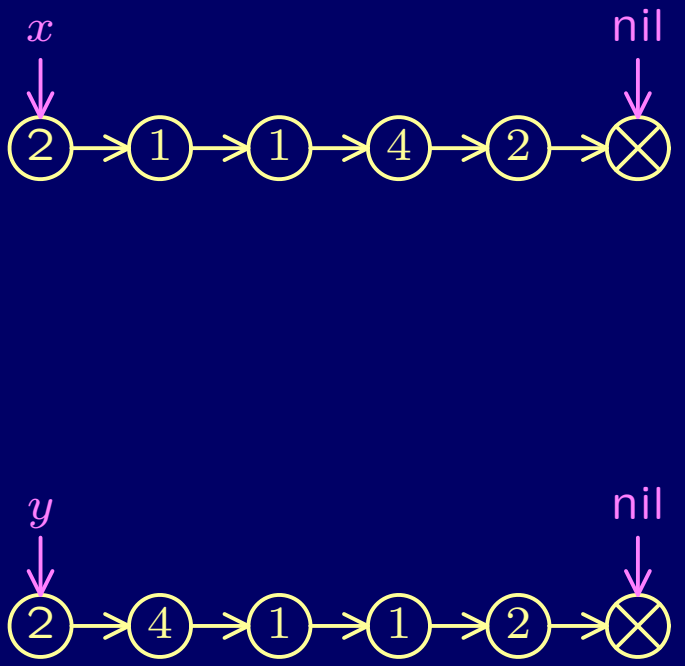
変数 nil は，常に特別なセルを指すものとする.

操作記述

- 変数への代入． `x := nil;`
- ポインタをたどる．`x := y.next.next.next;`
- 整数値を書きかえる．`x.val := 3;`
- ポインタを張りかえる．`x.next := y;`
- 制御文:
  - `if (x == nil) {...}`
  - `while (x.val != 0) {...}`

検証イメージ

```
//(1)
y := nil;
while (x != nil) {
  t := y;
  y := x;
  x := x.next;
  y.next := t;
}
//(2)
```



(1): x **から** nil **に到達する．**[nil $\wedge$ $\mathbf{EF}x$ **あり**]

(1): x **の後ろでは，**1 **の次に** 2 **は来ない．**[$2 \wedge \mathbf{EX}(1 \wedge \mathbf{EF}x)$ **なし**]

(2): y **の後ろでは，**2 **の次に** 1 **は来ない．**[$1 \wedge \mathbf{EX}(2 \wedge \mathbf{EF}x)$ **なし**]

ツール構築のステップ

- 操作と述語を記述する方法
- 操作ごとの最弱前条件を計算する手続き
- 時相論理式の充足可能性を決定する手続き
- モデル検査器とのインタフェース

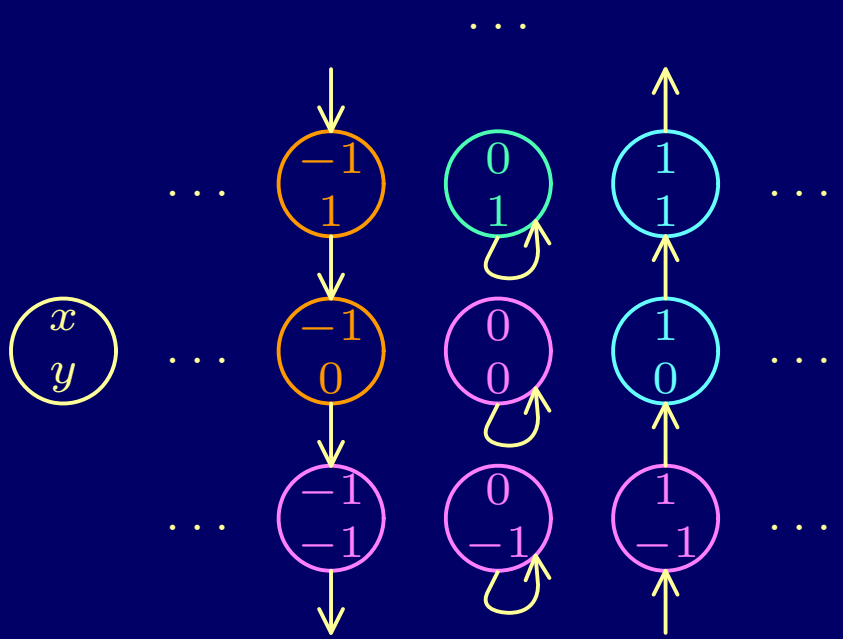
（スライド末尾）

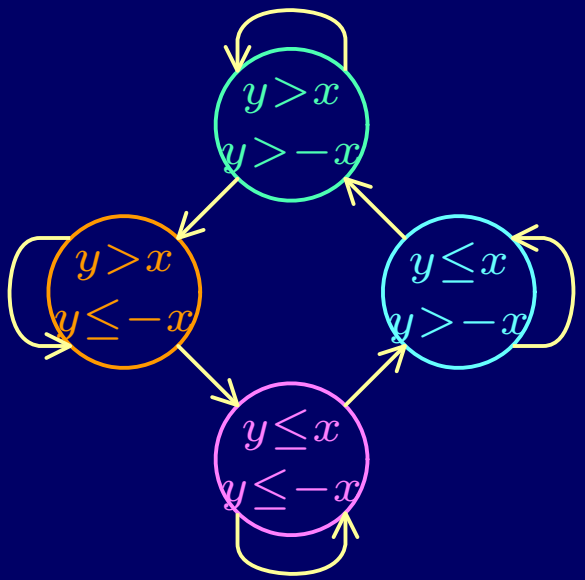
述語抽象化

具体モデル ⟶ **抽象モデル**

述語集合

$\{\text{“}y > x\text{”},\ \text{“}y > -x\text{”}\}$



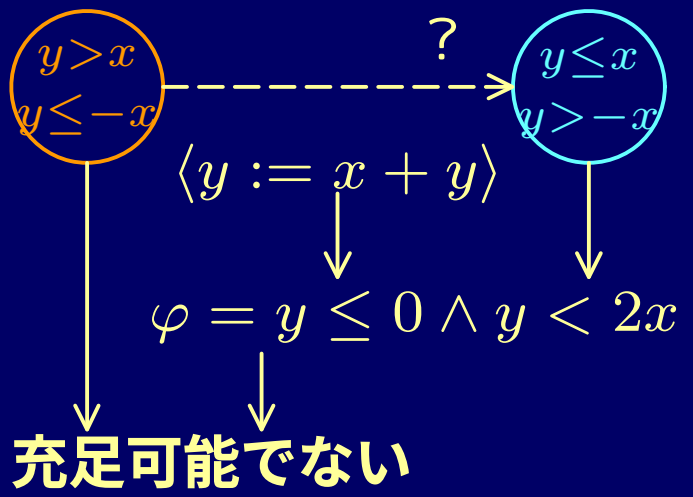


($y := x + y$ **による遷移**)

抽象遷移の決定

$\{P_1, \neg P_2\}$ から $\{\neg P_1, P_2\}$ に遷移するか?

- 具体モデルでの遷移に関して，$\neg P_1 \wedge P_2$ の最弱前条件 φ を求める．
- $\varphi \wedge P_1 \wedge \neg P_2$ が充足可能であるかどうかを判定する．
- Yes なら，遷移がある．
  No なら遷移しない．



ツール化には，充足可能性を判定する機能が重要．