

# 自動証明系と対話型証明支援系の 連携によるポインタ操作プログラムの 検証について

(MLAT と Agdaの接合)

湯浅 能史(産総研) 武山 誠(産総研)

関澤 俊弦(産総研・阪大) 田辺 良則(産総研・東大)

高橋 孝一(産総研)

# 対話証明器 vs 自動証明器

- 対話型証明支援系
  - 長所：“あらゆる”性質が証明/反証可能。
  - 短所：人手による証明。手間がかかる。
- 自動定理証明器
  - 長所：機械任せ。“手間がかからない”。
  - 短所：示せる性質は限定的。

連携による検証が有効

# Agda + MLAT

- Agda ---- 対話型証明支援系。Martin-Löf型理論に基づく。Emacs による ユーザーインターフェース。
  - MLAT ---- 抽象遷移系生成器。ポインタ操作プログラムが対象。様相論理を用いた述語抽象。
- 

- 連携の方法 ---- Agdaプラグイン機構を利用。
- 検証の枠組み ---- Hoare 論理。
- 検証の対象 ---- ポインタ操作プログラム。手続き型。

# 目次

- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- Pml-Hoare 論理
- 抽象遷移系生成器 MLAT について
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- Agda-MLAT連携による検証例
  - リスト反転プログラムの検証
  - ループ不変式の確定
- まとめ

# 目次

- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- Pml-Hoare 論理
- 抽象遷移系生成器 MLAT について
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- Agda-MLAT連携による検証例
  - リスト反転プログラムの検証
  - ループ不変式の確定
- まとめ

# Agdaによる定理証明

$$\frac{\frac{(\wedge\text{-ER}) \frac{[A \wedge B]^{ab}}{B}}{\quad} \quad \frac{[A \wedge B]^{ab}}{A} (\wedge\text{-EL})}{\quad} (\wedge\text{-I})}{\quad} (\rightarrow\text{-I})$$
$$A \wedge B \rightarrow B \wedge A$$

```
comCnj (A,B :: Set) :: A ∧ B -> B ∧ A
= ∀(ab :: A ∧ B)->
  and elimAndR ab
  elimAndL ab
```

Emacs

- Curry-Howard 対応に基づいた証明の符号化。
- 関数型プログラミング言語に似た証明記述。
- 対話的にゴールをサブゴールに展開する。

# Agdaによる定理証明

ab :: A ∧ B

$$\begin{array}{c} \frac{\frac{(\wedge\text{-ER}) \frac{[A \wedge B]^{ab}}{B}}{A} \quad \frac{(\wedge\text{-EL}) \frac{[A \wedge B]^{ab}}{A}}{B}}{B \wedge A} (\wedge\text{-I})}{A \wedge B \rightarrow B \wedge A} (\rightarrow\text{-I}) \end{array}$$

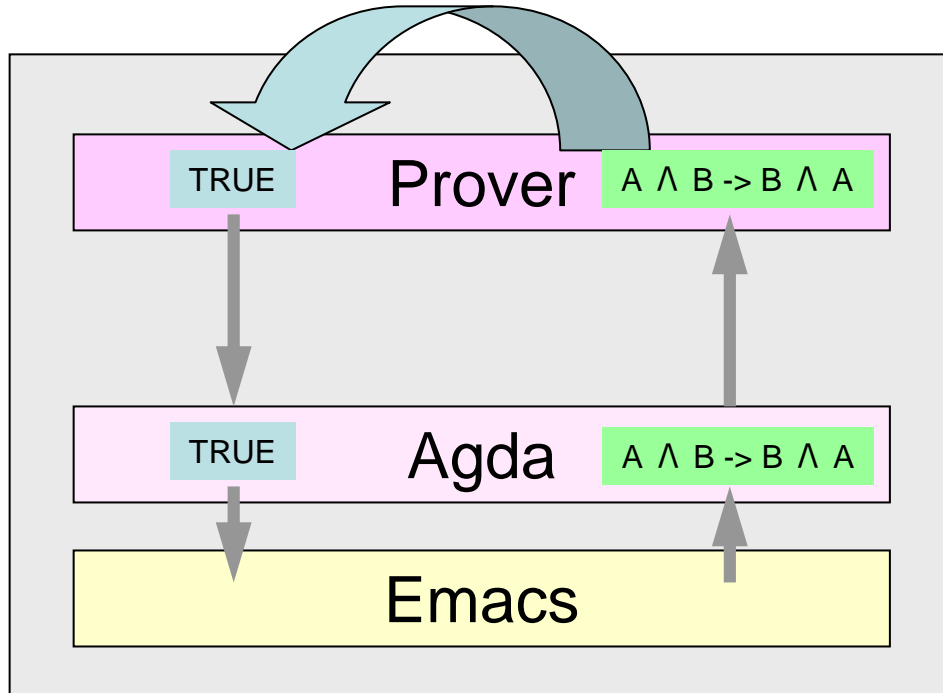
```
comCnj (A,B :: Set) :: A ∧ B -> B ∧ A
```

```
≡ λ (ab!) : A ∧ B -> {! !}  
  λ a b! → λ a! b! →  
    λ a! b! → {! !}
```

Emacs

- Curry-Howard 対応に基づいた証明の符号化。
- 関数型プログラミング言語に似た証明記述。
- 対話的にゴールをサブゴールに展開する。

# Agda Prover Plug-in



```
comCnj (A,B :: Set) :: A & B -> B & A
= {external\FoFol"!
```

Emacs

- 既存の自動証明器に手を加えることなく、Agda側の設定でだけで接続できる。
- これまで接続した自動証明器：
  - gandalf (FOL)
  - NuSMV



# 目次

- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- **Pml-Hoare 論理**
- 抽象遷移系生成器 MLAT について
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- Agda-MLAT連携による検証例
  - リスト反転プログラムの検証
  - ループ不変式の確定
- まとめ

# PML-Hoare 論理

Pointer Manipulation Language

- 基本条件式 AS :  $x == \text{Null}$ ,  $\frac{y}{f}$  Hoare Triple (Hoare 式)
- 基本命令文 AC :  $x := \text{Null}$ ,  $x.$   $P \{cs\} Q$   $\bullet$   
 $x := \underline{y.f}$ ,  $x.$   $cs$  実行前に  $P$  なら 実行後には  $Q$   
ノード  $y$  の  $f$ -フ...
- プログラム

条件式 :  $S ::= AS \mid (!S) \mid (S \&\& S) \mid (S \parallel S)$

$\dots$  else  $\{CS\}$  | while  $S$  do  $\{CS\}$

最弱前条件:

命令  $c$  実行前に  $wpc \ Q$  ならば実行後には  $Q$

Agda上に”深い符号化”で実現

$wpc \ Q$  は上記を満たす中で、最弱のもの

述べる論理式 Pformula (後述)

$$\frac{P \Rightarrow wpc \ Q}{P \{c\} Q} \text{ (atm)} \quad (\text{for } c \in AC)$$

$$\frac{P \wedge \bar{s} \{cs_0\} Q \quad P \wedge \neg \bar{s} \{cs_1\} Q}{P \{if \ s \ \text{then} \ \{cs_0\} \ \text{else} \ \{cs_1\}\} Q} \text{ (ite)}$$

$$\frac{P \Rightarrow R \quad R \wedge \bar{s} \{cs\} R \quad R \wedge \neg \bar{s} \Rightarrow Q}{P \{\text{while } s \ \text{do} \ \{cs\}\} Q} \text{ (whl)}$$

$$\frac{P \Rightarrow Q}{P \{\lambda\} Q} \text{ (nil)}$$

$$\frac{P \{e\} R \quad R \{cs\} Q}{P \{c; cs\} Q} \text{ (cbs)}$$

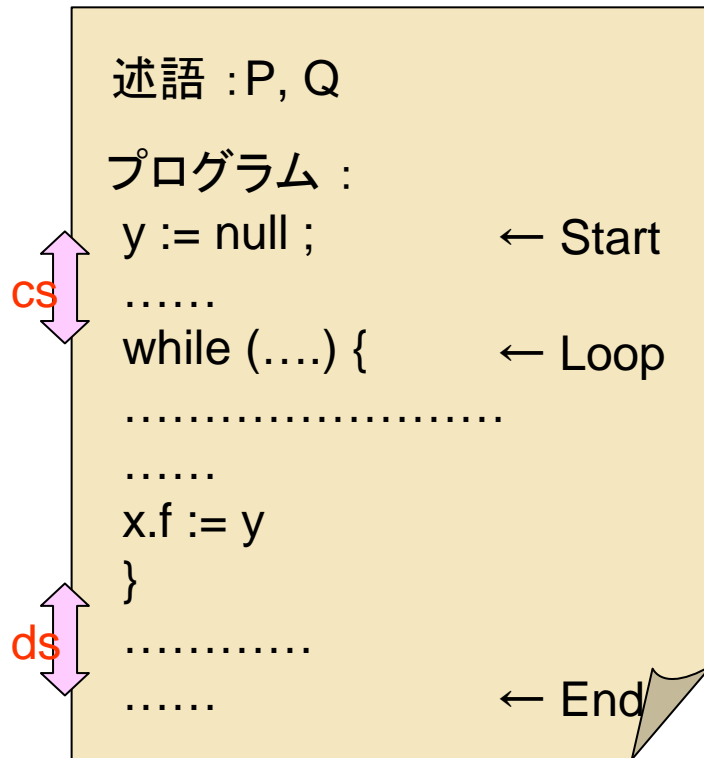
(制限付き)

MLATで自動証明

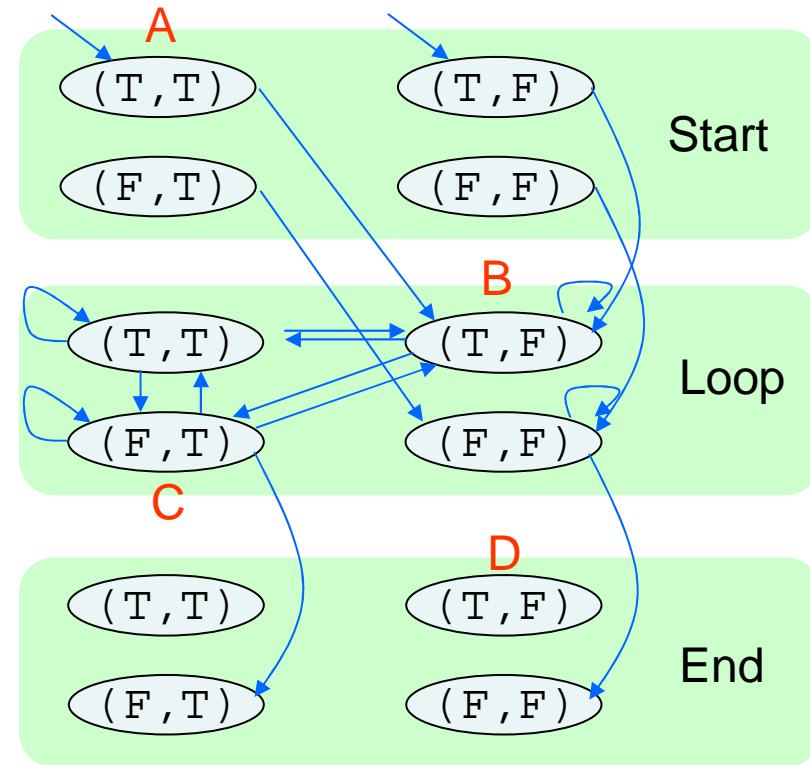
# 目次

- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- Pml-Hoare 論理
- **抽象遷移系生成器 MLAT について**
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- Agda-MLAT連携による検証例
  - リスト反転プログラムの検証
  - ループ不変式の確定
- まとめ

# MLAT (Modal Logic Abstraction Tool)



抽象遷移系の作成



「最弱前条件(wp)」と「充足可能性判定(sat)」により遷移可能性を判定する。

$\text{sat}(A \wedge \text{wp}(\text{cs}, B)) = \text{true} \Rightarrow$  遷移あり

$\text{sat}(C \wedge \text{wp}(\text{ds}, D)) = \text{false} \Rightarrow$  遷移なし

$$\text{pfvalid}(X, \text{cs}, Y) = \text{not sat}(X \wedge \text{wp}(\text{cs}, \text{not } Y))$$

Hoare式の判定

# ポインタ構造と時相論

フィールド  
様相

プログラム実行のある時 でのヒープの状態を リブ構造とみなす  
性質を 様相CTL式で記述する。

変数 : x, y, .....

フィールド: f, g, .....

プログラム:

y := null ;

.....

while (.....) {

.....

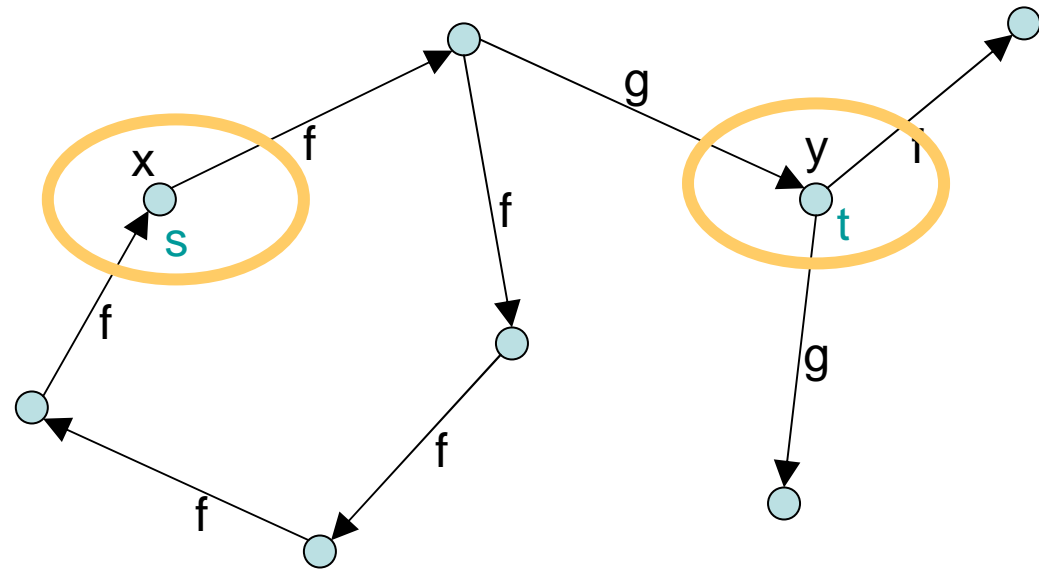
.....

x.f := y

}

.....

.....



変数 x がノード s を指している:  $\exists x \exists s (s \neq \text{null} \wedge x.f = s)$

変数 y がノード t を指している:  $\exists y \exists t (t \neq \text{null} \wedge y.g = t)$

# PML-Hoare 論理

- 基本条件式 AS :  $x == \text{Null}, x.\text{val} == b, x == y$

- 基本命令文 AC :  $x := \text{Null}, x.\text{val} := b, x := y,$

Pformula:

式 " $x \vdash P$ " の命 合 ( $x$ : プログラム変数,  $P$ : CTL式)

例:  $(x \vdash P) \Rightarrow (y \vdash Q) \wedge (z \vdash R)$

- プログラム

条件式 :  $S ::= AS \mid (S) \mid (S \&\& S) \mid (S \mid S)$

命令文 :  $C ::= AC \mid \text{if } S \text{ then } \{CS\} \text{ else } \{CS\} \mid \text{while } S \text{ do } \{CS\}$

プログラム :  $CS ::= \lambda \mid C, CS$

Agda 上に "深い符号化" で実現

- 推論規則 ( $P, Q, R$ : ヒープの状態を記述する論理式)

$$\frac{P \Rightarrow \text{wp}_c Q}{P \{c\} Q} \text{ (atm)} \quad (\text{for } c \in \text{AC})$$

$$\frac{P \wedge \bar{s} \{cs_0\} Q \quad P \wedge \neg \bar{s} \{cs_1\} Q}{P \{\text{if } s \text{ then } \{cs_0\} \text{ else } \{cs_1\}\} Q} \text{ (ite)}$$

MLATで自動証明

$$\frac{P \Rightarrow R \quad R \wedge \bar{s} \{cs\} R \quad R \wedge \neg \bar{s} \Rightarrow Q}{P \{\text{while } s \text{ do } \{cs\}\} Q} \text{ (whl)}$$

$$\frac{P \Rightarrow Q}{P \{\lambda\} Q} \text{ (nil)}$$

$$\frac{P \{c\} R \quad R \{cs\} Q}{P \{c; cs\} Q} \text{ (cns)}$$

(制限付き)

# 目次

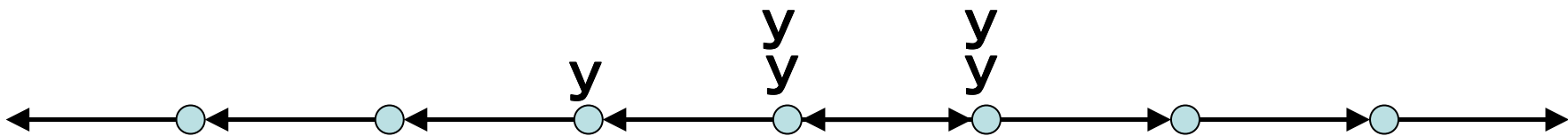
- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- Pml-Hoare 論理
- 抽象遷移系生成器 MLAT について
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- **Agda-MLAT連携による検証例**
  - リスト反転プログラムの検証
  - ループ不変式の確定
- まとめ

# 検証例「リスト反転プログラム」

reverse :

```
y := Null ;  
while (not (x == Null)) do {  
  t := y ;  
  y := x ;  
  x := x.next ;  
  y.next := t ;  
} ;
```

- ループを一回すると：
- ・ t, y, x が 1つ移動
  - ・ リストが1つ反転





# Agdaで

Listp x := (x |- EF NULL)

x から NULL (= nil )に する

前/ 後条件

```
Pre :: PFormula = ListP x && Reach x u && Next u v && not (Nullp v)
Pos :: PFormula = Next v u
```

ループ不変式

Inv ::

A

$$\frac{\text{Inv} \Rightarrow \text{Inv} \quad \text{Inv} \wedge \neg \text{Nullp } x \{ \dots \} \text{Inv} \quad \text{Inv} \wedge \neg \text{Nullp } x \Rightarrow \text{Pos}}{\text{Inv} \{ \text{while}(x \neq \text{Null}) \text{do} \{ \dots \} \} \text{Pos} \quad \text{Pos} \Rightarrow \text{Pos}}$$

$$\frac{\text{Pre} \{ y := \text{Null} \} \text{Inv} \quad \text{Inv} \{ \text{while}(x \neq \text{Null}) \text{do} \{ \dots \} \} \text{Pos}}{\text{Pre} \{ \text{reverse} \} \text{Pos}}$$

Hoare

rev :: (HI Pre reverse Pos)

MLATによる自動証明

= let lem1 :: HTc Pre (setNull y) Inv = external "mlat"

lem2 :: HTp (Inv && not (Nullp x)) loopBody Inv = external "mlat"

lem3 :: PFImp (Inv && not (not (Nullp x))) Pos = external "mlat"

in cns\_lem1 (cns\_ (whl\_id lem2 lem3) (nil\_id\_))

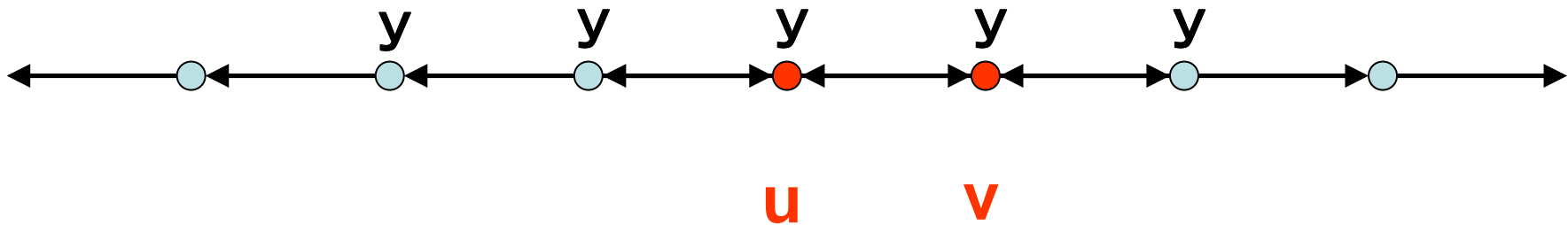
# ループ不変式の確定

1st trial

```
ListP x && not (Nullp v) &&  
  ( Reach x u && Next u v ||  
    Equal x v && Equal y u ||  
    Next v u )
```

ループ不変式

→ “No” ( MLAT )



# ループ不変式の確定

3rd trial

```
ListP x && not (Nullp v) &&
```

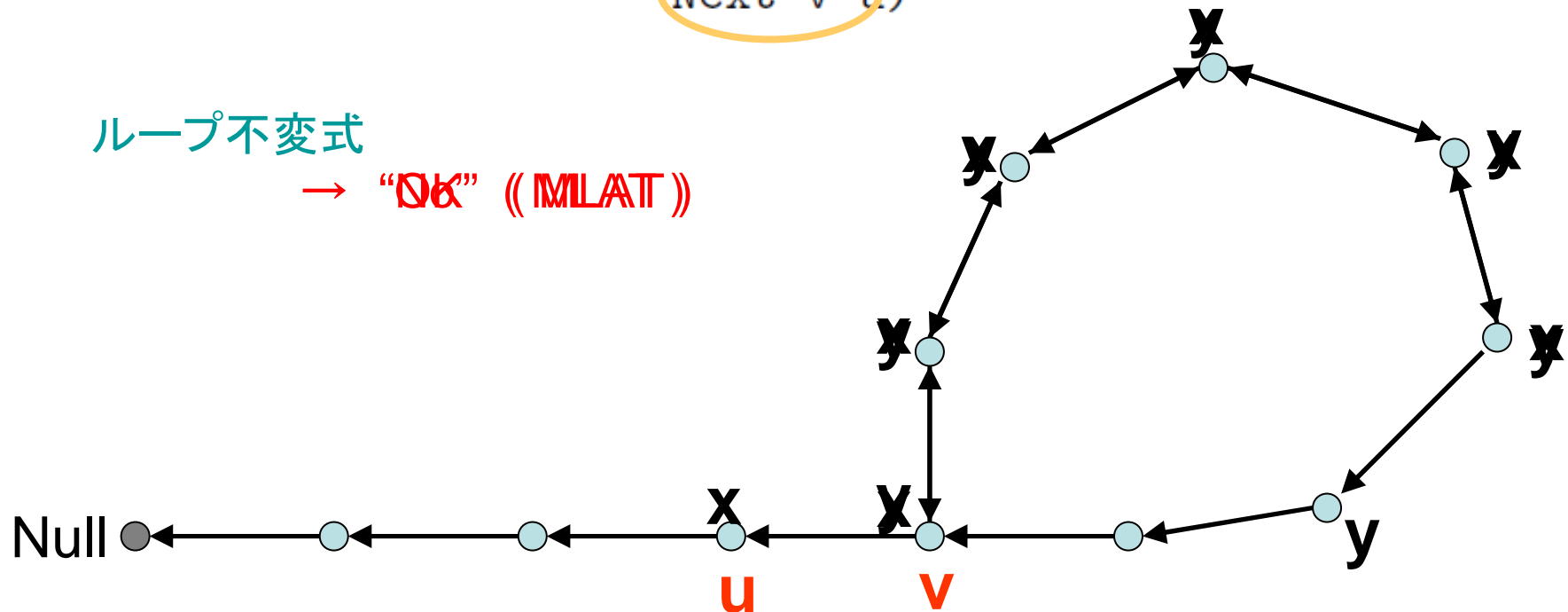
```
( Reach x u && Next u v ||
```

```
Equal x v && Equal y u ||
```

```
Next v u.)
```

ループ不変式

→ “OK” ((MLAT))



# Agdaでの証明記述

```
Pre :: PFormula = ListP x && Reach x u && Next u v && not (Nullp v)
Pos :: PFormula = Next v u
```

ループ不変式

```
Inv :: PFormula = ListP x && not (Nullp v) &&
  Agda-MLAT連携による Reach x u && Next u v ||
  行 で not (Reach x u) && Equal x v && Equal y u ||
  not (Reach x u) && Next v u
```

```
rev :: HT Pre reverse Pos
  = let lem1 :: HTc Pre (setNull y) Inv = external "mlat"
      lem2 :: HTp (Inv && not (Nullp x)) loopBody Inv = external "mlat"
      lem3 :: PFImp (Inv && not (not (Nullp x))) Pos = external "mlat"
  in cns_ lem1 (cns_ (whl_ id_ lem2 lem3) (nil_ id_))
```

MLATによる自動証明

# 目次

- 対話証明器 Agda について
  - Agda による定理証明
  - Agda Prover Plug-in
- Pml-Hoare 論理
- 抽象遷移系生成器 MLAT について
  - MLAT の仕組み
  - ポインタ構造と時相論理式
- Agda-MLAT連携による検証例
  - リスト反転プログラムの検証
  - ループ不変式の確定
- **まとめ**

# まとめ: と 後の

自動-対話 の リト

- 対話証明器を用いた により、自動証明にかかる時間を短 できる。( 0s → 10s)
- 行 の として自動証明器を用いることができる。(ループ不変式の な )

後の

- 最弱 前条件の 。(現状では、Hoare式の証明は MLATでしか できない)
- MLATからAgda の の け し。( のときに反例、 のときに証明の ヒント)
- 大きな例 、 の 、 etc.

り