

分散プログラムを 形式的証明から抽出する

東京大学大学院情報理工系研究科
萩谷研究室 平井洋一

2009-09-17

大会論文の修正版:

- ▶ 余りは空き机に置いてください。
- ▶ セッション後に平井が回収します。

ということによろしいでしょうか。

問題: 分散プログラムを 正しく書くのは困難

分散プログラムを書き間違えても、
なかなか気付かないことがある

- ▶ 百万回に一回のバグ

本研究の対象: wait-free プログラム
プロセス間同期無し．実分散的．

解決法: プログラムでなく 形式的証明を書く

形式的証明を書き間違えると、
すぐに気付く

- ▶ 証明チェッカ

本研究の対象: 知識論理の変種
プロセスの知識を記述できる。

仕様

が書いてある

を満たす

証明

抽出

Wait-free
プログラム

Wait-free プログラム

[Herlihy99] の , wait-free protocol.
他のプロセスを待てない

できないこと

入力 プロセス a, b に , 0 か 1 か
出力 プロセス a と b の両方が ,
相手の入力を出力する .

Wait-free プログラムの例

入力 プロセス a, b に, 0 か 1 か
出力 少なくとも片方が, 相手の
入力を出力する .

```
*shm_x = *shm_y = -1; /* 共有メモリの初期化 */  
/* 以下並列実行 */
```

```
int a(int x) {  
    *shm_x = x;  
    if (*shm_y != -1)  
        return *shm_y;  
    else  
        return -1;  
}
```

```
int b(int y) {  
    *shm_y = y;  
    if (*shm_x != -1)  
        return *shm_x;  
    else  
        return -1;  
}
```

*shm_x と *shm_y は別 . Sequential consistency を仮定

仕様

 $\Gamma \vdash_G \Delta$

Γ 実行前に全部成立と仮定

Δ 実行後に一つ成立と保証

G 実行に關与するプロセス

Γ, Δ は論理式の有限列

$$\varphi ::= P \mid K_a \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

$K_a \varphi$ プロセス a が φ を知ってる

\wedge, \vee かつ, または

仕様の例

$$K_a P \wedge K_b Q \vdash_{\{a,b\}} K_b K_a P \vee K_a K_b Q$$

P であることを a が知っていて、
 Q であることを b が知っている **ならば**
 $\rightarrow a, b$ の wait-free プログラムで \rightarrow
 Q であることを b が知っていることを
 a が知っているか、
 P であることを a が知っていることを
 b が知っている **ようにできる**

証明: 実現可能な仕様を網羅する

$$\text{(axiom)} \frac{}{\varphi \vdash_G \varphi} \quad \text{(cut)} \frac{\Gamma \vdash_G \Delta, \varphi \quad \varphi, \Gamma' \vdash_G \Delta'}{\Gamma, \Gamma' \vdash_G \Delta, \Delta'} \quad \text{(K)} \frac{}{K_a \varphi \vdash_{\{a\}UG} \varphi}$$

$$\text{(comm)} \frac{}{K_a \varphi, K_b \psi \vdash_{\{a,b\}UG} K_a K_b \psi, K_b K_a \varphi}$$

$$\text{(nec)} \frac{\Delta \vdash_G \Gamma}{K_a(\bigwedge \Gamma) \vdash_{GU\{a\}} K_a(\bigvee \Delta)} \quad (\text{上のほうに (comm) が無い})$$

$$\text{(K}\wedge\text{)} \frac{}{K_a \varphi, K_a \psi \vdash_G K_a(\varphi \wedge \psi)} \quad \text{(K}\vee\text{)} \frac{}{K_a(\varphi \vee \psi) \vdash_G K_a \varphi, K_a \psi}$$

$$\text{(}\vee\text{ 左)} \frac{\Gamma, \varphi \vdash_G \Delta \quad \Gamma', \varphi' \vdash_G \Delta'}{\Gamma, \Gamma', \varphi \vee \varphi' \vdash_G \Delta, \Delta'} \quad \text{(}\vee\text{ 右)} \frac{\Gamma \vdash_G \Delta, \varphi, \varphi'}{\Gamma \vdash_G \Delta, \varphi \vee \varphi'}$$

$$\text{(}\wedge\text{ 左)} \frac{\Gamma, \varphi, \varphi' \vdash_G \Delta}{\Gamma, \varphi \wedge \varphi' \vdash_G \Delta} \quad \text{(}\wedge\text{ 右)} \frac{\Gamma \vdash_G \Delta, \varphi \quad \Gamma' \vdash_G \Delta', \varphi'}{\Gamma, \Gamma' \vdash_G \Delta, \Delta', \varphi \wedge \varphi'}$$

証明の例

Wait-free プログラムの抽出

(comm) \mapsto

各プロセスが、共有メモリに書
いてから読む

(cut), (\vee 左), (\wedge 右) \mapsto

プロセスごとの接続 ;

各変数への代入が一度だけになる
ように、変数をたくさん使う。

示したこと

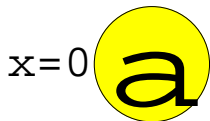
- ▶ 健全性:

仕様に証明がある ⇒
仕様をみたす wait-free プログラムが有る

[Herlihy99] による wait-free 計算の特徴づけで使われた, 単体的複体上に意味論をつくった(続く).


意味論の概要(1)

$x = 0$ でプロセス a が起動



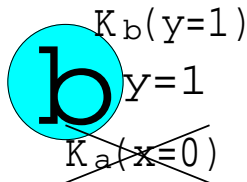
意味論の概要 (2)

$x = 0$ はプロセス a の知識

$x=0$ 
 $K_a(x=0)$

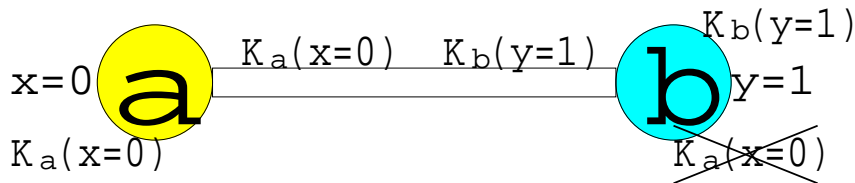
意味論の概要 (3)

$y = 1$ でプロセス b が起動
プロセス a の知識は, わからない



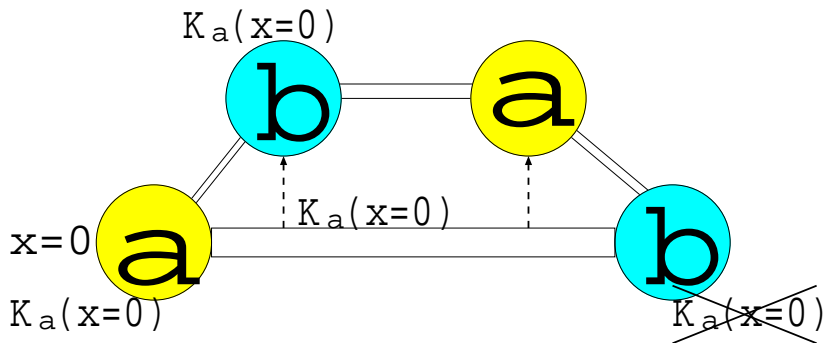
意味論の概要 (4)

プロセス a, b の両方の状態が見える視点からは、少なくとも片方で成立する命題は成立する



意味論の概要 (5)

Wait-free protocol は単体的複体の分割である [Herlihy99] .

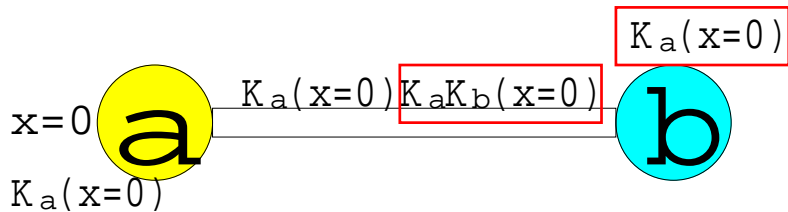


大会論文の修正理由: つまらない話だった

- ▶ 意味論が，もっとゆるかった
- ▶ 証明体系も，ゆるかった
- ▶ 超健全性が成立: 実現可能な任意の仕様を「何もしない」プログラムが満たす
- ▶ 抽出しても嬉しくない

大会論文の修正理由: 意味論がゆるすぎる

hold, but not wanted



Future work

- ▶ 完全性を示す

仕様に証明がある \Leftarrow

仕様満たす wait-free プログラム有

- ▶ 関連研究との比較:

動的知識論理 [Ditmarsch07],

単体的複体上の知識論理 [Porter04]

- ▶ [Attiya04] の wait-free 計算を扱う
(while 文が入る)

Future future work: もっと弱く

Sequential consistency が成り立たない場合を調べる

- ▶ 計算を scale させたい .
- ▶ 公理を弱くして , cut 除去可能な論理を得たい
(仕様を保ちプログラムを短く変形できる)

Future future work: **もっと強く**

プリミティブを様相として追加する機構を作る

- ▶ **「ここだけは同期が必要」と証明中に明示する**

参考文献

- [Herlihy99] Herlihy, M. and Shavit, N.: The Topological Structure of Asynchronous Computability, *J. ACM*, Vol. 46, No. 6, 1999, pp. 858–923.
- [Attiya04] Attiya, H. and Welch, J.: *Distributed Computing Fundamentals, Simulations, and Advanced Topics, 2nd Edition*, Wiley, 2004.
- [Ditmarsch07] Ditmarsch, H. v., Hoek, W. v. d., and Kooi, B.: *Dynamic Epistemic Logic*, Springer, 2007.
- [Porter04] Porter, T: Interpreted systems and Kripke models for multiagent systems from a categorical perspective, *Theor. Comput. Sci*, Vol. 323, No. 1–3, 2004, pp. 235–266.