SEAMAIL

Newsletter from Software Engineers Association

Vol. 12, Number 8-10 May, 2001

目　次

編集部から		1
フォーマル・メソッドとは何か？	二木 厚吉	3
開発におけるフォーマル・メソッドの適用	荒木 啓二郎	30
オブジェクト指向のための形式手法 -- 振舞仕様とモデル検査による自動検証について	森 彰	41
形式手法のAからZまで：Zをどう教えたらよいか	玉井 哲雄	73
洗練計算	山崎 利治	86
Java とその拡張の形式化：形式化を通じての言語デザインにかんする考察	五十嵐 淳	119
フォーマルメソッドのコンポーネントソフトウェア開発への適用について	松本 充広	152
ソフトウェア工学に於いて「数学的」とはどういう事か？ －精密ソフトウェア工学に向けて －	金藤 栄孝	189
応答系の仕様記述	山崎 利治	221

ソフトウェア技術者協会

Software Engineers Asociation

ソフトウェア技術者協会 (SEA) は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985年 12月に設立されました．

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約 200人にすぎなかった会員数もその後増加し，現在，北は北海道から南は沖縄まで，500 余名を越えるメンバーを擁するにいたりました．法人賛助会員も 25社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，広島，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます．SEA は，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

代表幹事： 玉井哲雄

常任幹事： 荒木啓二郎 坂本啓司 高橋光裕 田中一夫 中野秀男 深瀬弘恭

幹事： 伊藤昌夫 大場充 落水浩一郎 窪田芳夫 熊谷章 小林修 桜井麻里
酒匂寛 塩谷和範 篠崎直二郎 新谷勝利 新森昭宏 杉田義明 武田淳男
中来田秀樹 野中哲 野村行憲 野呂昌満 端山毅 平尾一浩
藤野誠治 松原友夫 山崎利治 和田喜久男

事務局長： 岸田孝一

会計監事： 橋本勝 吉村成弘

分科会世話人 環境分科会(SIGENV)：塩谷和範 田中慎一郎 渡邊雄一
教育分科会(SIGEDU)：君島浩 篠崎直二郎 杉田義明 中園順三
ネットワーク分科会(SIGNET)：人見膚 松本理恵
プロセス分科会 (SEA-SPIN))：伊藤昌夫 坂本啓司 高橋光裕 田中一夫 端山毅 藤野誠治

支部世話人 関西支部：臼井義美 小林修 中野秀男 横山博司
横浜支部：野中哲 藤野晃延 北條正顕
長野支部：市川寛
名古屋支部：筏井美枝子 石川 雅彦 角谷裕司 野呂昌満
九州支部：武田淳男 張漢明 平尾一浩
広島支部：大場充 佐藤康臣 谷純一郎
東北支部：河村一樹 布川博士 野村行憲 和田勇

賛助会員会社：ジェーエムエーシステムズ 東芝アドバンストシステム ＳＲＡ ＰＦＵ
東電ソフトウェア 構造計画研究所 富士通 オムロンソフトウェア
キヤノン 中央システム 富士通エフ・アイ・ピー 新日本製鉄
ダイキン工業 東北コンピュータ・サービス オムロン ＳＲＡ中国
NECソフト 富士電機 ブラザー工業 オリンパス光学工業 リコー
アルテミスインターナショナル NTTデータ ヤマハ 福井コンピュータ （以上25社）

ＳＥＡＭＡＩＬ Ｖｏｌ．１２，Ｎｏ．８-１０ ２００１年5月２１日発行
編集人 熊谷章
発行人 ソフトウェア技術者協会（ＳＥＡ）
〒１６０-０００４ 東京都新宿区四谷３－１２ 丸正ビル５Ｆ
T: 03-3356-1077 F: 03-3356-1072 sea@sea.or.jp
印刷所 有限会社 錦正社 〒１３０-００１３ 東京都墨田区錦糸町４－３－１４
定価 １，５００円 （禁無断転載）

編集部から

関東では一足飛びに初夏がやってくる。三週間程前まで桜の花と新芽で目を楽しませてくれた自然がゴールデンウィークが過ぎればすっかり様変わりしてしまう。葉桜を見やりながら、つぼみや満開そして花を落とした直後の紅いガクが桜の木一面を覆ったころを思い出すのも一興である。桜の花といえば今年は変わった風景に遭遇した。

4月 17、18、19 日と石川に行ったのだ。春たけなわなので、電車が一番と考え、越後湯沢、十日町、富山、金沢と旅窓を楽しむことにした。越後湯沢は積雪 30 センチほどで桜が咲いていた。雪どけの鉄砲水が水路から溢れ、山肌には残雪があり、桜が処処に咲いている風景に出会ったのである。遠くには、八海山や越後駒ケ岳が青い空に聳え、全体として幽玄な雰囲気だ。「21 世紀に残したい風景」として NHK に投書したいと思った。

湯沢からホクホク線で金沢に向かう。この線の乗客はなぜかみなさん裕福そうな顔をしている。そう考えればなぜか自然と楽しくなる。左右の窓から見える景色は早春そのもの。まんさくの花やふきのとう、それに野蒜があるのが遠目にも分かる。春のシンフォニーが聞こえて来るようだ。

直江津に出ると目の前に日本海が開けてくる。海を見ればいつも海の色は空の色を写しているに過ぎないという事を思い出さずにいられない。猟師が一人佇んでいる海岸は誰もいない海より寂しい。　親不知では芭蕉を思い出した。空中に高く浮いているような高速道路を座標にして眺める日本海は奇妙な感じがした。

糸魚川辺りだと左手に妙高山、雨飾山、雪倉岳、白馬岳が白く輝いているのを見れる。春で空気が霞んでいるため白い山並がまるで蜃気楼のように見える。黒部川は雪解け水をたたえてゆっくり流れている。遠くには槍や穂高と思える山並がみえる。.滑川では、立山と毛勝山を見ることができる。この辺りは、越後湯沢と違ってとうに葉桜で、時折八重桜の満開を見付けることができる頃である。鎌倉とほぼ一緒の気候だと思った。砺波ではチューリップをまだ見ることはできなかった。春まだきなのだろう。
東京発 12 時で金沢着4時半の電車の旅です。機会があれば試してみてはいかがでしょうか。勿論、酒肴付きでないといけないのはいうまでもありません。

今回は、sigfm 特集号です。2000 年 3 月の発足会からの一年間の活動の中からいくつかのトピックスを紹介します。今後のフォーマルメソッドの振興と発展の契機の一つにしたいと思いを込めたものです。様々な分野やいろいろな所でフォーマルメソッドが使用される時期が間もなくやってくるのは一目瞭然です。次ページに活動履歴をご紹介致します。今後も皆様のご支援とご協力を切にお願い申し上げます。(くま)

2000年3月3日 発足記念フォラム

1部　フォーマルメソッドの現状　司会：佐原伸 (SRA)

「フォーマルメソッドと CafeOBJ 」二木厚吉 (JAIST)

「開発におけるフォーマルメソッドの適用」荒木啓二郎 (九州大学)

2部　パネル討論「フォーマルメソッドをいかに普及させるか？」司会: 熊谷章 (PFU)

パネラ:山崎利治 (Free)、金藤栄孝(日立) 、伊藤昌夫(ニルソフト) 、上記発表者

2000年4月7日　定例会

「クリーンルーム法の概略と事例」染谷 誠 (日本ユニシス)

2000年5月12日　定例会

「オブジェクト指向ソフトウェア開発法 Catalysis の概要と制約記述法について」

伊藤 昌夫 (ニルソフトウェア)

2000年6月2日　定例会

「オブジェクト指向のための形式手法： 振舞仕様とモデル検査による自動検証について」　森 彰 (北陸先端科学技術大学院大学(JAIST))

2000年7月7日　定例会

「形式手法の A から Z まで：Z をどう教えたらよいか」玉井 哲雄(東京大学)

2000年9月8日　定例会

「洗練計算」と「非正則集合の話」山崎利治（フリー）

2000年10月31日　SIGFM Special in Ishikawa

「Java とその拡張の形式化：形式化を通じての言語デザインにかんする考察」

五十嵐 淳（東大）

「フォーマルメソッドのコンポーネントソフトウェア開発への適用について」

松本充広（PFU, JAIST)

2000年12月12日　定例会

「事務処理システムにおける形式仕様記述言語(VDM++)の適用事例研究ーシステムの概要、アーキテクチャについてー」　酒匂寛 (デザイナーズ デン)

「具体的な記述の内容と、チャレンジ 」谷津弘一 (日本ユニシス)

2001年1月11日　定例会

「M-pi 計算: モバイルコードと分散コンピューティング」富樫敦(静岡大学)

2001年3月9日　定例会

「ソフトウェア工学に於いて「数学的」とはどういう事か？ー精密ソフトウェア工学に向けて ー」金藤栄孝（日立製作所システム開発研究所）

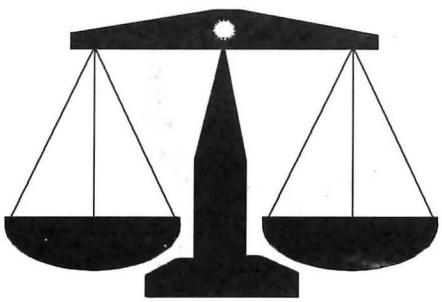
2001年4月6日　定例会

「鉄道に於けるコンピュータ制御鉄道信号システム」平尾（鉄道総合技術研究所）

「ＶＤＭを利用した連動装置シミュレータ」福田（鉄道総合技術研究所）

「Fizgerald 先生の講義スライドの説明」荻野（鉄道総合技術研究所）

フォーマルメソッドとは何か？



北陸先端科学技術大学院大学
JAIST
二木 厚吉
FUTATSUGI, Kokichi

この講義でお話すること

- フォーマルメソッドとは何か？
  - ソフトウェア開発における形式化の意義と限界
- フォーマルメソッドの利点と欠点
- 色々なフォーマルメソッド，言語，ツール
- フォーマルメソッドをどう役立てるか？
- CAFE, CafeOBJとは何か？

フォーマルメソッドとは何か？

フォーマルメソッドとは何か？(1)

- フォーマルメソッド
- （formal methods：形式手法）
  - 論理学や離散数学などに基礎をおくソフトウェア開発法
  - 要求，仕様やプログラムを集合、関数、代数といった数学モデルに基づいて作成する.
    - -> 形式仕様(formal specification)

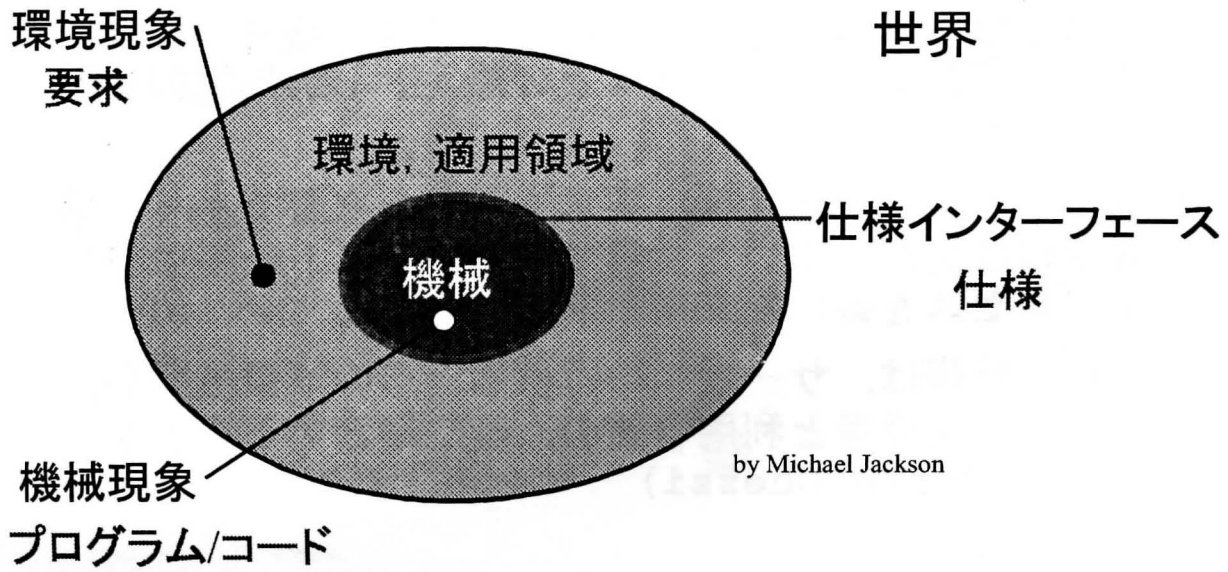
仕様とは何か？ (1)

- 仕様とは？「特定の形状，構造，寸法，成分，精度，性能，製造法，試験方法などの規定」[岩波国語辞典(第四版)]
- 文書，例えば要求仕様は要求を書いた文書であり，設計仕様は設計を書いた文書であり，テスト仕様はテストプロセスを書いた文書である．(Alan Davis)
- 一般に仕様は，サービスの提供者とその消費者間の，あるいは開発者と利用者間の，契約書と見ることができる．(Carlo Ghezzi)

仕様とは何か？ (2)

- 仕様は何を(what)を記述する
  - 問題領域(application domain)
- プログラムはいかに(how)を記述する
  - 設計，実装，機械(Machine)，アルゴリズム

仕様とは何か？ (3)



けいしき【形式・型式，form】とは？ (1)

（広辞苑第４版）（form） 事物の内容に対し、外から認められるものとしての形。特に、個々の形に重点を置く場合と、通じて見られる型に重点を置く場合とがあり、また，内容と切り離していることを強調する場合がある。⇔内容。

- (1) うわべに現れた形。外見。外形。（時として、実質を問わないという非難のニュアンスがある） 「論文の―を成していない」「通り一遍の―だけで済 ます」

けいしき【形式・型式, form】とは？ (2)

- (2)事務などを進めるための、文書の体裁や執るべき手続。「この届の—は不備だ」「—を踏んで面会を求める」
- (3)芸術作品の美的効果を高めるように、作品の構成する諸部分を配合する仕方。
- (4)〔哲〕物事の材料・内容と切り離して、構造・型・枠組を抽象してえられるもの。哲学上の概念としては形相(けいそう)と訳す。

けいしき【形式・型式, form】とは？ (3)

(American Heritage Dictionary) formal adj.

- 1.a. Relating to or involving outward form or structure. b. Being or relating to essential form or constitution.
- 2.a. Following or being in accord with accepted forms, conventions, or regulations. b. Executed, carried out, or done in proper or regular form.
- 3.a. Characterized by strict or meticulous observation of forms; methodical. b. Stiffly ceremonious.
- 4. Having the outward appearance but lacking in substance.

ソフトウェア開発と形式化

- ソフトウェア開発とは何か？，という問いに対する答えは人により異なり．明確ではない．
- ソフトウェア開発，とくに要求と仕様に関する基本概念に対する明快な説明がMichael Jacksonの近著 "Software Requirement & Specifications a lexicon of practice, principles and prejudices"(玉井，酒匂訳：ソフトウェア博物誌)に見られる．
- これは，ソフトウェア開発における形式化の役割と限界についての最良の解説でもある．

非形式から形式へ -- 問題領域から機械へ

- 非形式性を形式化する際の難しさは本物である．それに失敗することによって招かれる以上のひどいあやまちは考えられない．この作業をうまくやることが，良いシステムを構築する際に必要な作業の大きな(しばしば最大で最も重要な）部分を占めるのである．
（Michael Jackson; 玉井，酒匂訳）

形式化の前提

- 適用領域中の概念の明確化と，用語の定義.
  - 用語: ものとその上の操作の名前
  - Michael Jacksonの指示(designations)
    - 適用領域と機械の間の局所的な狭い橋
  - 形式仕様の指標(signature)も同じ目的を持つ！

何について話しているのか分からないのに，きっちりやろうなんて無意味だ.

概念がはっきりしていなかったり，適用すべき対象の問題がはっきりしていないときに，正確な方法を使っても仕方ない.
(John von Neumann)

モデルの種類とその形式化

Ackoffの科学モデルの分類(1962)

- 図像(iconic)モデル
  - 模型飛行機
- 擬似(analogic)モデル -- 形式モデル
  - 液体の流量制御のための電気回路モデル
- 分析(analytic)モデル -- 形式モデル
  - 経済分析のための微分方程式モデル
  - フォーマルメソッドは，ソフトウェア開発における要求と仕様のための，数学化/形式化に基づく分析モデルを提供する

計算機に対する基本的認識

計算機は『形式』を操作する機械である
(機械は形式のみを操作する)

ソフトウェア科学・工学は
「内容/適用領域」を「形式/機械」に還元するための
モデル・理論・技術・手法を明らかにする

フォーマルメソッドとは何か？(2)

- 数学的に形式化された構文と意味を持つ形式仕様言語 (formal specification language) を用いたシステム開発法

フォーマルメソッドは仕様レベル,
つまり要求と仕様の,
形式化/数学化を目指す
(従来はプログラム/ソフトウェアの形式化であった;
is it a prejudice?)

要求/仕様と数学

- ソフトウェア開発者にとって，数学は単なる道具である．そのすべてではなく，大きな部分を占めるものでもなく，一番重要なものでもない．ソフトウェア開発の中心的な課題は(特に要求と仕様においては)証明や論理的演繹ではない．むしろ関心事の分割と，真であることを明らかにする明確性である．また数学的抽象化ではなく，むしろ機械が仕事を行なう[問題の文脈]内の現象をきちんと区別し記述することが大切なのである．いずれにしろソフトウェア開発は計算ではない，記述なのである．(Michael Jackson; 玉井，酒匂訳)
  - software development is linguistic exercise
  - no formal methods for req&spec yet

記述の種類

- 粗いスケッチ(rough sketch)
  - あいまいな記述
  - 覚え書き，推測，
- 反駁可能な記述
  (refutable description; Karl Popper)
  - 真偽が客観的に決定できる記述
  - 形式的な記述
    - 形式言語で書かれた記述，等式，論理式
  - 論理実証主義 (A J Ayer)

仕様は反駁可能であり，無矛盾性、，完全性、非曖昧性、明晰性といった性質を持つことが望ましい

- 反駁可能(refutable)：真偽が客観的に決定できる記述
- 無矛盾性(consistent)：規定されている内容に自己矛盾がない.
- 完全性(complete)：必要な内容がすべて規定されている.
- 非曖昧性(unambiguous)：規定されている内容が一意に特定できる.
- 明晰性(transparent)：規定されるべき内容が理解し易く記述されている.

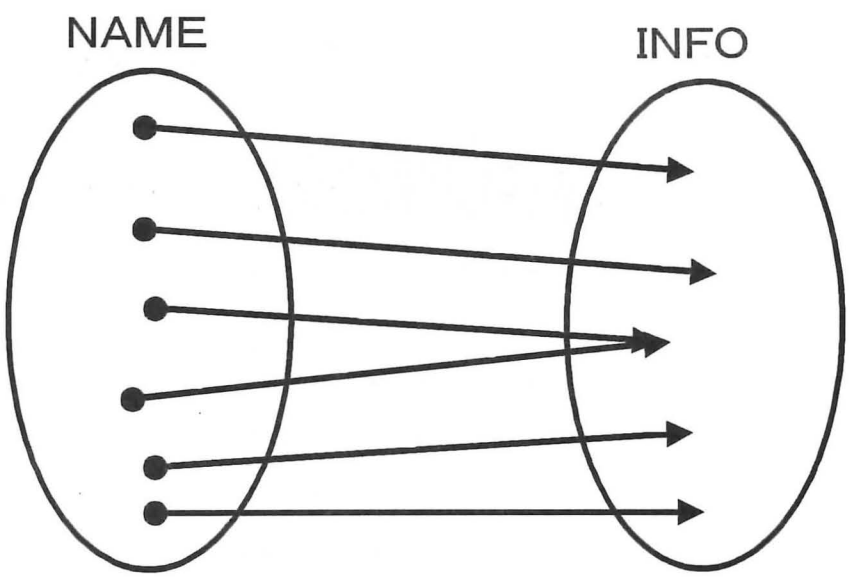
非形式仕様(Informal Specification)

- 自然語、図、ダイアグラムなどを用いて書かれた仕様のように，意味を定めるモデルや規則が与えられていない仕様
- 無矛盾性等の性質を判定する際の基準となるべき「仕様の意味」を客観的に同定することが難しく，反駁可能ではない
- 無矛盾性，完全性，非曖昧性などの性質を仕様が満たすかどうか判定することが困難である

形式仕様 (Formal Specification)

- あらかじめ与えられた規則に従い意味を特定することができる仕様
- 反駁可能であり，無矛盾性などの色々な性質を仕様のレベルで解析できる可能性が高い
- 計算機で処理しやすい
  - 形式言語(formal language)で書かれる

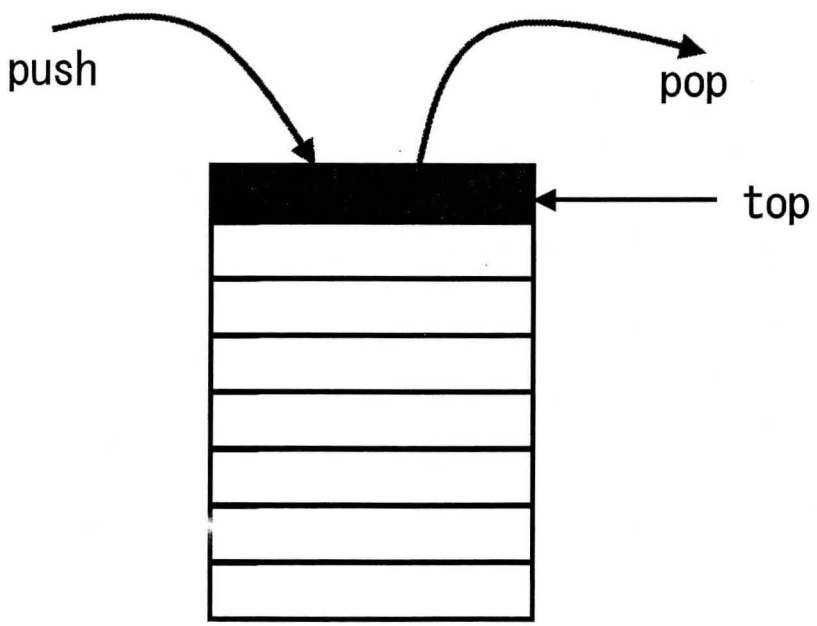
形式仕様の例 (1)
Zを用いたデータ辞書の仕様 - 図



形式仕様の例 (1)
Z記法を用いたデータ辞書の仕様 - コード

```
--- DataDictionary ----------------------
|  dict : NAME +-> INFO;
|  defined : P NAME
|---------
|  defined = dom dict;
|  # defined <= MaxSize
-----------------------------------------

DataDictInit is [ DataDictionary' | defined' = {} ]

--- InsertOk ----------------------------
|  Delta DataDictionary;
|  name? : NAME;
|  info? : INFO;
|  resp! : Response
|---------
|  name? notin defined;
|  # defined < MaxSize;
|  dict' = dict || { name? -> info? };
|  resp! = Success
-----------------------------------------
```


形式仕様の例 (2)
CafeOBJを用いたスタックの仕様 - 図



形式仕様の例 (2)
CafeOBJを用いたスタックの仕様 - コード

```
mod! STACK (X :: TRIV) {
        [ NeStack < Stack ]
        op nil : -> Stack          -- empty Stack
        op push : Elt Stack -> NeStack
        op pop_ : NeStack -> Stack  -- only applicable to NeStack
        op top_ : NeStack -> Elt    -- only applicable to NeStack

        var E : Elt
        var S : Stack

        eq top push (E, S) = E .
        eq pop push (E, S) = S .
}
```


フォーマルメソッドとは何か（３）？

- 形式仕様言語＋仕様記述法＋仕様解析法
  - 形式仕様言語の運用論(方法論)は，記述法(開発法)と解析法(検証法)を含む

- A way of formulating requirement and software development documents such that calculus can be made. (Dines Bjorner)

計算(calculus)または数学の特質

- 我々は数学的抽象化の大事な局面にさいかかった．個々の記号が何を表しているのかは忘れるのだ...怠けているわけではない．これらの記号を用いて数学者は実に多くの操作を行なうことが出来るのである．しかもその記号が表しているものを実際に見ることなしに．
(Herman Weyl)
- プログラム計算(program calculus)においては記号が何を表しているのを忘れて，計算のルールを間違い無く適用することに集中しなければならない．
(E.Dijkstra)

フォーマルメソッドの３要素

- 形式仕様言語 (notation, language)
- 形式仕様言語の意味モデル(意未定義: semantics)
- 形式仕様言語の運用法＝方法論 (methodology)
  - 記述法
  - 解析法，検証法

言語と方法論

- 方法論は適用領域に依存する
  - 問題フレームが方法論を規定する
    - さらに多フレーム問題が一般的
  - 卑しくもプロジェクトと呼ばれるような多く複雑なソフトウェアの開発のためには，吟味して仕立て直された複数の方法論の組が必要である．(Michael Jackson; 玉井，酒匂訳)
- フォーマルメソッドは汎用的でも万能でもない
  - 汎用性を持ち得るのは形式記法(notation)または形式言語(language)である
  - 記法，言語は多くの方法論に共通に使える
  - 通常フォーマルメソッドと呼ばれるものの多くは記法または言語である．-->厳密でない用語

形式仕様言語とプログラミング言語

- 形式仕様言語
  - =(人間の)思考の道具としての言語
- プログラミング言語
  - =(計算機への)命令を伝える道具としての言語
- 形式仕様言語は計算機言語の高級化の延長線上にある
  - 関数プログラミング，
  - 論理プログラミング，
  - 代数プログラミング

フォーマルメソッドの典型的なプロセス

- 形式仕様を作成する
- 仕様を解析してシステムの特性をチェックする
- 仕様を満たすように（段階的に）システムを開発する
  - 必ずしもトップダウン(top-down)開発ではない

ラッピドプロトタイピング(rapid prototyping)
拡張型開発法(incremental development)
Objectory Process:
Inspection, Elaboration,
Construction, Transition

フォーマルメソッドの利点と欠点

形式仕様の利点

- 仕様自体の機械処理が可能であり，仕様の解析，蓄積，再利用等に計算機を有効に利用できる可能性が高い.
- 形式仕様にはモジュール化/構造化を陽に指示する機能が用意されていることが多いので，仕様文書の構造化/モジュール化，再利用がし易い.

フォーマルメソッドの効用(1)

- システム開発の早い段階で仕様のチェックができる。
- 形式仕様を確定することで、システムの性質に対する理解が深まる。
  - 非形式仕様->形式仕様->非形式仕様
- システムの仕様の確定、開発、検証、維持管理が計算機の支援を得て系統的に行える。

フォーマルメソッドの効用(2)

- 再利用単位を特定しやすい。
- 高い信頼性を持つソフトウェアの開発が可能となる。
- 開発コストが軽減する。
  - 上流行程のコストは増加する
  - 下流行程のコストと保守管理のコストが軽減する

フォーマルメソッドの問題点

- 意味モデルから外れた概念の表現は不自然になり易く表現力，解析力が制限される．「現実とモデルの乖離」
- 形式仕様言語の構文規則，意味モデル，意味規則などの形式体系の学習を要求される「形式体系/モデルの教育(E.Djikstra)」
- 形式モデル，形式言語とその記述/解析法に知的集中を要求され，『問題への集中』が阻害されやすい．(はたしてほんとうにそうであろうか？良い道具を使えばより難しい問題を解決できるというのもまた真実である．)

技術的な課題(1)

- 自然言語化(広い意味のインターフェースの改善)
  - 形式仕様の自然言語化よりは，自然言語+形式仕様言語を使う方が有望であろう.
- 論理の拡張
  - プロセス代数、時制論理、ペトリネット...
- 領域固有性、世界観
  - 汎用モデルとドメインモデルの折衷

技術的な課題(2)

- 解釈領域の問題、動的特性の定式化
- OOとの関係？
  - UML, OOFSL, Design Methods using OOL
- 抽象化のレベルの混同
  - 理想的な解答はない
    - FMの実戦を通しての体系化、ライブラリ化
- 教育の問題
  - 逆説的に:FMは最も有効な教育手段である

色々なフォーマルメソッド，言語，ツール

フォーマルメソッド，言語の分類

- モデル化の視点： モデル指向、機能指向
- モデル化の単位： 状態、機能、プロセス、抽象データ型、オブジェクト
- モデル化の（数学的）道具： 集合、関数、論理、代数

"FM-languages and tools" from Oxfored-PRG-WWW-lib(1)

- The Z notation.
- VDM (Vienna Development Method). See also the VDM FORUM mailing list, the Mural tool, and VDM text books
- B-Tool from Edinburgh Portable Compilers Ltd., UK.
- B-Toolkit from B-CORE (UK) Ltd., UK.
- RAISE language and tools from CRI, Denmark.

"FM-languages and tools" from Oxfored-PRG-WWW-lib(2)

- OBJ - OBJ3 and 2OBJ.
- CafeOBJ
- CSP (Communicating Sequential Processes) including the FDR tool.
- The HOL mechanical theorem proving system. Isabelle.
- PVS (Prototype Verification System) tool based on classical typed higher-order logic developed at the SRI International Computer Science Laboratory.
- LP (Larch Prover). DEC SRC, CMU. The Larch tool set.
- EVES tool, based on ZF set theory, from ORA, Canada.

"FM-languages and tools" from Oxfored-PRG-WWW-lib(3)

- Nqthm theorem prover last updated in 1992.
- Developed by Boyer-Moore et al. at Computational Logic Inc. (CLInc), Austin, Texas, USA.
- Pi-calculus. (Also CCS.) See papers by Robin Milner et al.
- JAPE (Just Another Proof Editor) by Bernard Sufrin and Richard Bornat.
- ASLAN, a specification language processor/proof obligation generator at UC Santa Barbara.
- The LAMBDA toolset from Abstract Hardware Ltd, UK, supports formal verification for hardware/software co-design.

"FM-languages and tools" from Oxfored-PRG-WWW-lib(4)

- DisCo is specification method for reactive systems including a tool developed at the Tampere University of Technology, Finland.
- LOTOS (Language of Temporal Ordering Specifications) has a newsletter.
- Nuprl tool based on intuitionistic type theory.
- Mural tool to aid formal reasoning about specifications including a proof assistant and VDM support.
- TLA (Temporal Logic of Actions) tool support.

クリーンルーム手法

CLEANROOM PROCESS MODEL by Richard D. Linger, IEEE Software, Vol.11, No.2, March 1994, pp.50-58

- 段階的拡充型開発
  (Stepwise Incremental Development)
  - 開発と品質管理の分離
  - 関数概念(ボックス)に基く段階的詳細化と検証
  - フォーマルメソッドとしては古典的
  - 基本的にプログラムコードの設計/開発手法
  - 統計的テスト/品質保証

オブジェクト指向は フォーマルメソッドを必要とする

- オブジェクト指向設計とは、構造化された抽象データ型の集合として、ソフトウェアシステムを構築することである(Bertrant Meyer)
  - 方法論の基礎づけと形式化
  - クラスとオブジェクトの形式仕様の開発
  - オブジェクト指向システムの性質のチェック
  - 型チェック
  - 機能の整合性のチェック
  - 正しさのチェック/検証

フォーマルメソッドをどう役立てるか?

FMが有効な領域(1) -- 問題への集中

- 的確な意思疎通の道具
  - 自然言語、図、表との併用が有望
- 技術文書作成技術の延長上にある
  - system development is a linguistic exercise
- ドメインモデル構築の道具；問題への集中
  - 汎用モデル(component+function+behavior)
  - ドメイン依存モデル(バッチ処理、トランザクション処理、実時間システム
  - Michael Jacksonの問題フレーム(problem frame)

FMが有効な領域(2) -- 汎用性と信頼性

- software資産の再利用の手段
  - より上流行程における再利用
- 仕様(description)解析を通じての要求決定/設計の支援
  - design calculi (Z, B; VDM, RAISE; OBJ, CafeOBJ...)
- 高い信頼性を持つシステムを構築する最有力な手段
- 検証法、検証システム： 形式仕様無しでは意味が無い

いまなすべきことは何か？(1) -- 実践

- 形式仕様言語を用いた：
  - 中規模～大規模の応用領域のモデル化実検
  - 中規模～大規模の形式仕様開発実検

いまなすべきことは何か？(2) -- 道具の開発

- FM-CASEの開発
- OO-diagramとFormal Languageの融合
  - UMLの意味定義
- 領域を限定したFM-CASEの有効性の実証
  - M.Jacksonの問題フレーム
- オブジェクトベース/ライブラリ(in FSL)の整備
- 検証技術、検証システム整備
  - Safety, Security

フォーマルメソッドの展望

- FMの発展の２極化
- 意思疎通と文書化の道具/技術
  - 要求仕様、設計仕様文書
  - システムインターフェースの標準文書
    - ♥ 文書化->再利用->保守管理
- 高い信頼性が必要なシステムの開発技術
  - 超高信頼性システム
    - ♥ 通信、航空、鉄道、自動車、病院...
      - ○ 他の技術では代替できない
      - ○ 今日のhigh-techは明日のlow-tech

期待されるフォーマルメソッド

フォマルメッソドとは何か？ K.Futatsugi 53

Seamail Vol. 12 No. 8 - 10

SEA FORUM
フォーマル・メソッドの活用を目指して

開発における フォーマル・メソッドの適用

実用システムにおける事例報告

2000年3月3日

荒木啓二郎
（九州大学大学院システム情報科学研究科/
（財）九州システム情報技術研究所）

構成

- はじめに
  - 形式手法に対する認識
- 事例
  - 配電線監視制御システム
  - 自動倉庫システム
- 考察
  - 経過と効果
- おわりに
  - 今後の課題

はじめに

- 形式手法 (Formal Methods)に対する認識
  - J. A. Hall:
    Seven Myths of Formal Methods,
    IEEE Software, Vol.7, No.5, pp.11-19, 1990.
  - J.P. Bowen and M.G. Hinchey:
    Seven More Myths of Formal Methods,
    IEEE Software, Vol.12, No.4, pp.34-41, 1995.
- 日本では話題になることすらあまりない
  - Pre-Myths : 神話以前？

形式手法に関する七つの誤認識 (Hall, 1990)

- 形式手法はソフトウェアが完全であることを保証できる
- 形式手法とは須らくプログラムの証明である
- 形式手法はセーフティクリティカルシステムにのみ有効である
- 形式手法は高度に訓練された数学者を必要とする
- 形式手法は開発コストを増加させる
- 形式手法はユーザには受け入れられない
- 形式手法は現実の大規模ソフトウェアには使われない

誤認識への反論 (Hall, 1990)

- 開発の初期の段階での誤りの発見に有効
- 開発対象のシステム自体を深く考えさせることに寄与
- いかなる応用分野にも有効
- 数学を基礎としてはいるものの、プログラムよりは理解し易い
- 開発コストを減少させる
- 顧客の理解を助ける
- 産業界での実用プロジェクトに適用されて成功

研究経過

- 図式表現と形式仕様との融合
  - ERD、DFD、IDEF1X、UML : Z、VDM-SL
  - 4WD : シナリオ記述、システム分析・設計
- 関係データベースシステムから
  オブジェクト関係データベースシステムへ
  - IDEF1X ⇒ UML
- 仕様記述の事例研究
  - セキュリティプロトコルの形式仕様記述と検証:SSL
  - 配電線監視制御システム:モデル化、形式仕様
  - 自動倉庫システム:モデル化、形式仕様
- 多様で柔軟な記述:方法、道具建て、内容
  - Z, RAISE, VDM-SL, CCS, ML, ...

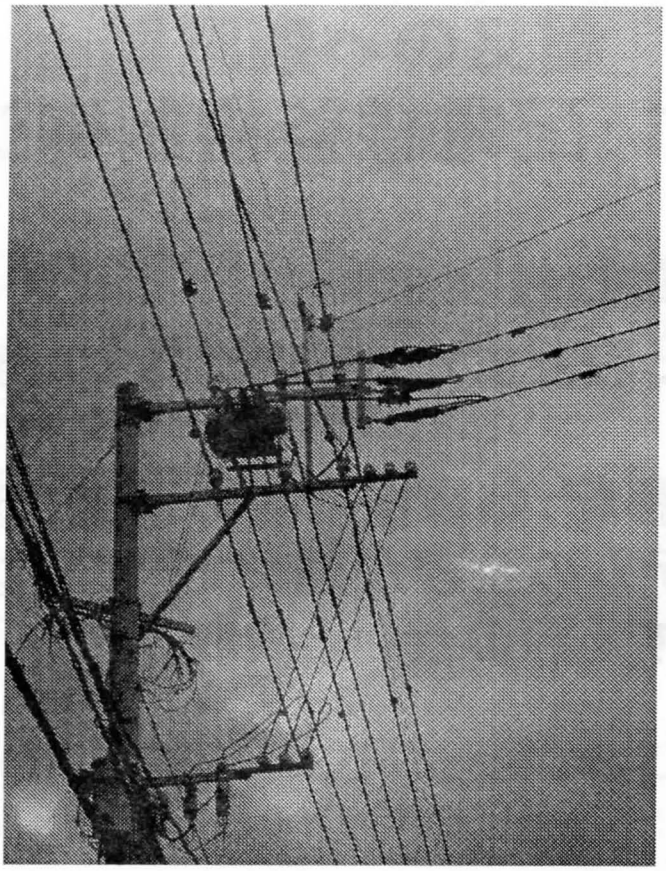
事例1:配電線監視制御システム

- 最初は自社開発
- 今は、設計までで実現は外部に発注

問題点
技術移転:熟練技術者の退職、新人教育、etc.
システムの全容が理解できていない
発注する際の仕様と検収

形式手法適用の目的

- リエンジニアリング
  - 汎用機からPCへ
- 再利用性
  - 異なる電力会社のシステムへの対応
- 技術移転/新人研修
- 社内文書の体系化
- システム開発の効率化
  - 高信頼性



開閉器（電線、電柱）

開閉器（地下ケーブル）

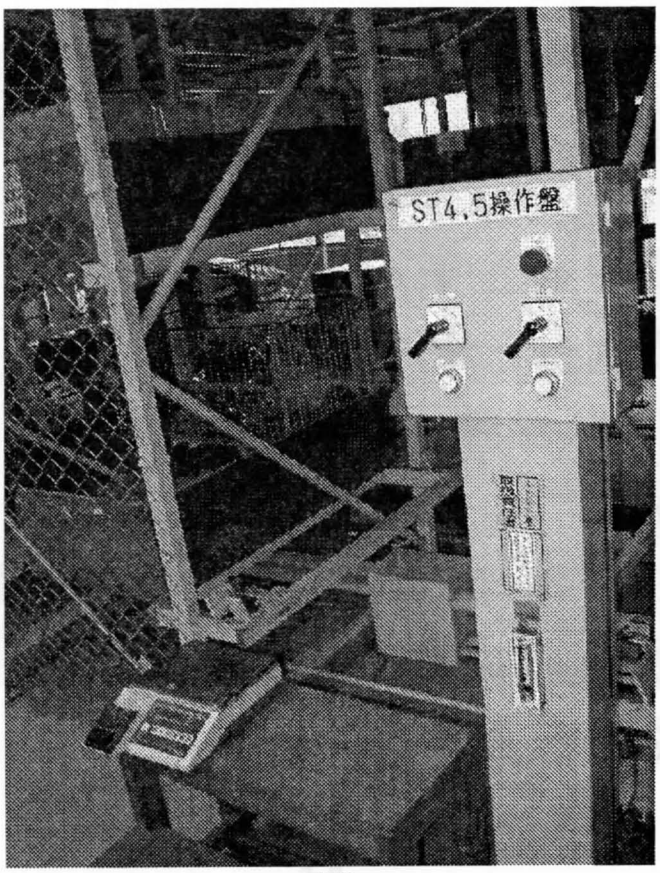
言語とツール(配電線)

- RAISE
  - モデル化
  - 仕様記述
- Z
  - 仕様の rewrite
- Java
  - プロトタイピング
- graphviz (AT&T Labs)
  - グラフ視覚化

事例2:自動倉庫システム

- 福岡の中堅企業
- 装置の制御とデータ管理
  - 設備(ラック、クレーン、搬送路、等)は別会社
- 多様性
  - 施設、設備、システム構成、用途、運用
- 再利用性
- 再構成の容易さ

自動倉庫（棚とクレーン）



自動倉庫（入出庫作業場）

言語とツール(自動倉庫)

- Z
  - モデル化
  - 仕様記述
- VDM-SL
  - 仕様の rewrite
- ML
  - プロトタイピング

形式手法適用の経緯

- 売込みと対象の選定
- Z のチュートリアル
- ドメインエキスパートから聞き取り
- 仕様記述の種々の版の作成と検討
  - 機能
  - モデル化
- 共通語としての Z
- アニメーション、プロトタイピング
  - ML, Java, graphviz

時間と労力

- 配電線システム
  - 約半年：月に2回程度会合
  - ドメインエキスパート：1～2名
  - 仕様記述とレビュー：4～6名
- 自動倉庫システム
  - 約半年：月に2回程度会合
  - ドメインエキスパート：1～2名
  - 仕様記述とレビュー：4～6名
- ドメインの理解とモデル/仕様記述の熟成

形式手法適用の効果

- コミュニケーションの道具
- 論点が明確
- ドメインの再認識

  暗黙の前提、常識：共通ではない
  曖昧性の顕在化

- 知識の共有、継承
  - 卒論生の理解と習得の早さ

形式仕様

- 読むのは、難しくない
  - 簡単な解説
  - 形式仕様のレビュー (OJT)
- 書くには、経験が必要
  - 言語仕様 (syntax)
  - 対象のモデル化 (semantics)
  - イディオム (pragmatics)
- アニメーション / 視覚化
  - multi-lingual
  - multi-aspect

おわりに

- 有効性は確認できた
  - システムの一部
  - モデル化と仕様記述
  - アニメーション、プロトタイピング
- 設計と実現
  - stepwise refinement ?
- 保守
- 文書

- 管理職の意識
  - 形式手法への関心、期待
  - 担当ドメインエキスパートの意欲への影響
- 厳密な記述と議論
  - 最初の戸惑い
  - 新鮮な感動
  - 慣れ
  - 維持継続の努力と支援
- コミュニティ
  - 普段の相談相手
  - guru

- コミュニケーション
  - 分析者
  - 仕様記述者
  - 設計者
  - プログラマ
  - 保守
  - 利用者
- ソクラテスの対話法
  - 概念の明確化
  - 用語の定義

オブジェクト指向のための形式手法

～振舞仕様とモデル検査による自動検証について～

森 彰
北陸先端科学技術大学院大学

1

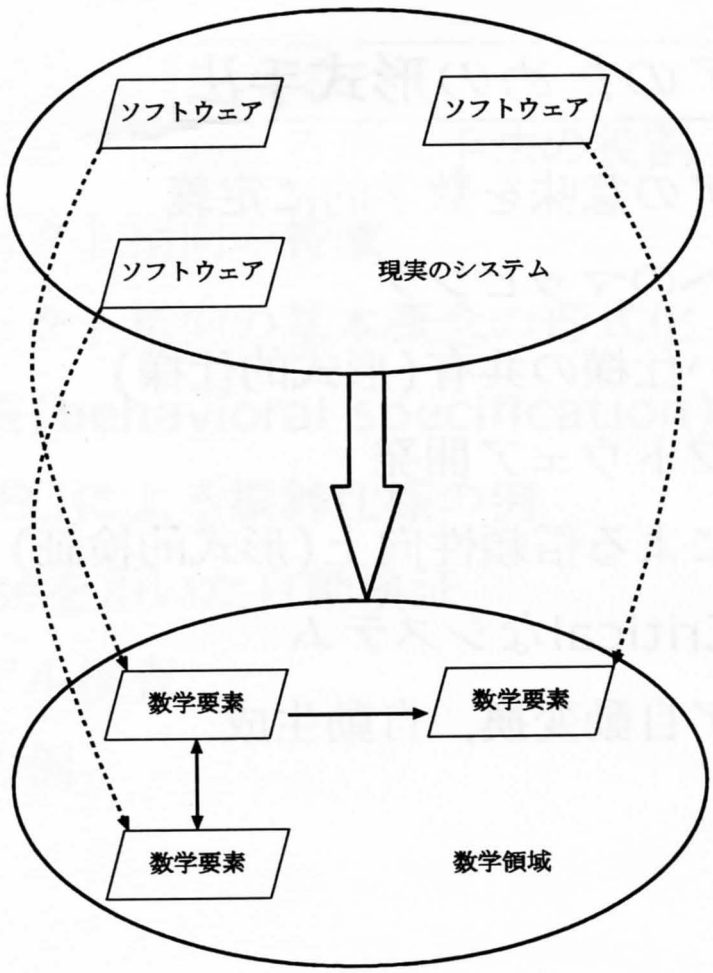
1. ソフトウェアにおける形式手法の役割
2. オブジェクト指向の特徴
3. オブジェクト指向の基本概念の形式化
4. 振舞仕様(behavioral specification)
5. CafeOBJによる振舞仕様の例
6. PigNoseを用いた自動検証
7. 振舞モデル検査
8. その他の例

2

ソフトウェアのための形式手法

- ソフトウェアの意味を数学的に定義
  - 数学領域へのマッピング
- 曖昧さのない仕様の共有(形式的仕様)
  - 大規模ソフトウェア開発
- 正当性検証による信頼性向上(形式的検証)
  - Safety-Criticalなシステム
- ソフトウェア自動変換，自動生成
  - !!???

3



4

代表的な手法

- プログラム論理 − ホーア論理，動的論理，...
- 表示的意味論 − 型理論，高階論理，構成的数学，...
- 代数仕様 − OBJ, Larch, ASD/SL, ...
- プロセス代数 − CSP, CCS, ACP, ...
- 集合仕様 − Z, VDM, B, ...
- 時相論理 − LTL, CTL, TLA, ...
- 状態機械 − I/O オートマトン，ペトリネット，...

5

プログラミング言語における抽象化の進化

言語(例)	対象	抽象化機構	形式手法
機械語	該当なし	該当なし	該当なし
アセンブリ言語	ラベル	番地抽象	フローチャート
FORTRAN	制御文	制御抽象	帰納関数論
PASCAL	サブルーチン	手続き抽象	ホーア論理
HASKEL	関数	関数抽象	高階型理論
CLU	データ型	データ抽象	代数仕様
Java	クラス	クラス抽象	???
ADL	アーキテクチャ	???	???

6

問題点

- 仕様と実装のギャップ
- 難解な数学(論理，普遍代数，カテゴリ論)
- 自動検証の困難さ
- スケールアップの困難さ

限られた応用

- 小規模アルゴリズムの正当性検証
- Safety-criticalなシステムで部分的に利用
- 領域に依存した特殊な利用

7

オブジェクト指向

- モジュール化，カプセル化
  - データ抽象＋内部状態
  - 情報隠蔽(アクセス制御)
- 再利用性
  - 継承
  - 抽象クラス
  - 動的結合による多相性

8

オブジェクト指向の基本概念の形式化

1. シグネチャ

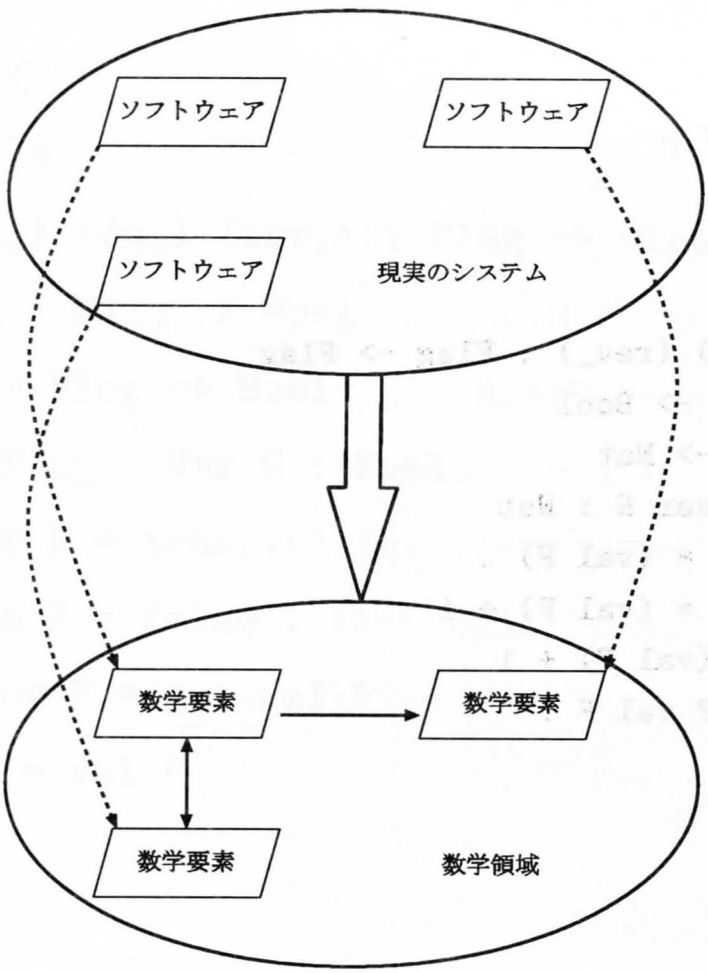
```
mod* ACCOUNT {
  protecting(INT)
  *[ Account ]*
  op new-account : -> Account              -- new account
  bop balance : Account -> Int             -- attribute
  bop deposit : Int Account -> Account     -- method
  bop withdraw : Int Account -> Account    -- method
}

public class Account {
  public int balance
  public void deposit(int amount) { ... }
  public void withdraw(int amount) { ... }
}
```

9

情報隠蔽に関する代数仕様の問題点

- 情報隠蔽(抽象化)が十分でない

10



11

FLAGオブジェクト

```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag
  bop up?_ : Flag -> Bool
  op val_ : Flag -> Nat
  var F : Flag   var N : Nat
  eq val up F = 2 * (val F) .
  eq val dn F = 2 * (val F) + 1 .
  eq val rev F = (val F) + 1 .
  eq up? F = even? val F .
 }
```

12

```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag
  bop up?_ : Flag -> Bool
  op val_ : Flag -> Bool
  var F : Flag   Var B : Bool
  eq val up F = true .
  eq val dn F = false .
  eq val rev F = not val F .
  eq up? F = val F .
}
```

13

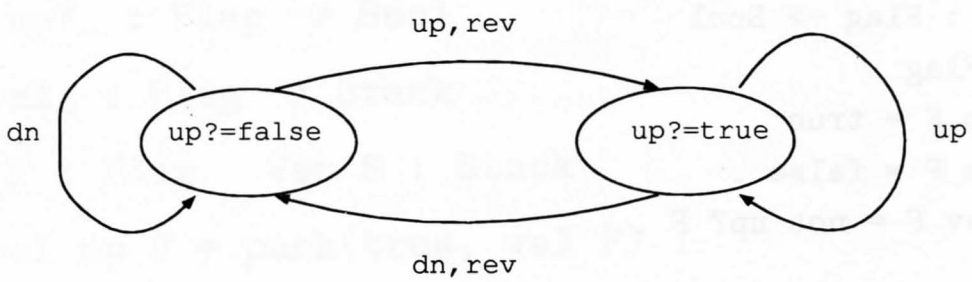
```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag
  bop up?_ : Flag -> Bool
  op val_ : Flag -> Stack
  var F : Flag   Var S : Stack
  eq val up F = push(true, val F) .
  eq val dn F = push(false, val F) .
  eq val rev F = push(not up? F, val F) .
  eq up? F = top val F .
 }
```

14

```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag
  bop up?_ : Flag -> Bool
  var F : Flag
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
 }
```

rev rev F = Fを導くことができるか？

15

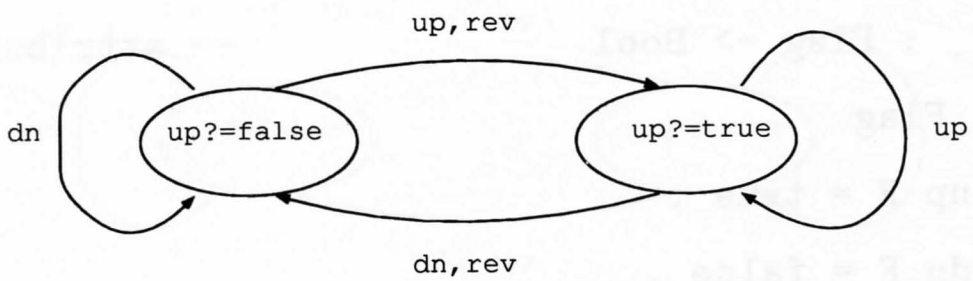


16

```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag  -- methods
  bop up?_ : Flag -> Bool                 -- attribute
  var F : Flag
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
}
```

$$\texttt{rev rev F} = \texttt{F?}$$

17


up,rev
dn
up?=false
up?=true
up
dn,rev

Hidden algebra as coalgebra

$$(\mathcal{G}(S))_h = (\prod_{\sigma: wh \to h'} [D^{|w|} \Rightarrow S_{h'}]) \times (\prod_{\sigma: wh \to v} [D^{|w|} \Rightarrow D^{|v|}])$$

where $S = S_{h(h \in H)}$.

18

Temporal Structure for Hidden Algebra

- Predicate as set of states

```
P(S:Protocol)=
∀I,J:Nat flag(I,S)=flag(J,S)=shared ⇒
                cdata(I,S)=cdata(J,S).
```

- Predicate transformers as "next time" operators

$$\widetilde{\mathtt{pre}}(\mathtt{P}(\mathtt{X}:\mathtt{h})_{\mathtt{h}})_{\mathtt{h}'} \stackrel{\triangle}{=} \prod_{\sigma:\mathtt{wh}'\to\mathtt{h}} (\forall\, \mathtt{V}:\mathtt{w})\ \mathtt{P}(\sigma(\mathtt{V},\mathtt{Y}:\mathtt{h}'))_{\mathtt{h}}$$

$$\mathtt{pre}(\mathtt{P}(\mathtt{X}:\mathtt{h})_{\mathtt{h}})_{\mathtt{h}'} \stackrel{\triangle}{=} \sum_{\sigma:\mathtt{wh}'\to\mathtt{h}} (\exists\, \mathtt{V}:\mathtt{w})\ \mathtt{P}(\sigma(\mathtt{V},\mathtt{Y}:\mathtt{h}'))_{\mathtt{h}}$$

(duality) $\widetilde{\mathtt{pre}}(\mathtt{P}) = \neg\mathtt{pre}(\neg\mathtt{P})$, $\mathtt{pre}(\mathtt{P}) = \neg\widetilde{\mathtt{pre}}\neg\mathtt{P}$

19

(Cont...)

- Galois connections (adjunction in $\mathcal{POSET}$)
  - $\langle L, \subseteq \rangle \xrightarrow{\alpha} \langle M, \sqsubseteq \rangle$ and $\langle M, \sqsubseteq \rangle \xrightarrow{\gamma} \langle L, \subseteq \rangle$
  - $\alpha(P) \sqsubseteq Q \Leftrightarrow P \subseteq \gamma(Q)$ (denoted by $\alpha \dashv \gamma$)
- "Last time" operators

$$\widetilde{\mathtt{post}}(\mathtt{P}(\mathtt{X}:\mathtt{h})_{\mathtt{h}})_{\mathtt{h}'} \triangleq \prod_{\sigma:\mathtt{wh}\to\mathtt{h}'} (\forall\, \mathtt{V}:\mathtt{w})\ \mathtt{Y}:\mathtt{h}' = \sigma(\mathtt{V},\mathtt{X})$$

$$\mathtt{post}(\mathtt{P}(\mathtt{X}:\mathtt{h})_{\mathtt{h}})_{\mathtt{h}'} \triangleq \sum_{\sigma:\mathtt{wh}\to\mathtt{h}'} (\exists\, \mathtt{V}:\mathtt{w})\ \mathtt{Y}:\mathtt{h}' = \sigma(\mathtt{V},\mathtt{X})$$

(duality) $\widetilde{\mathtt{post}}(\mathtt{P}) = \neg\mathtt{post}(\neg\mathtt{P}),\ \mathtt{post}(\mathtt{P}) = \neg\widetilde{\mathtt{post}}\neg\mathtt{P}$

20

(Still cont...)

- $\mathtt{post} \dashv \widetilde{\mathtt{pre}}$ and $\mathtt{pre} \dashv \widetilde{\mathtt{post}}$ are Galois connections
- Fixpoint operators

  least fixpoint μ and greatest fixpoint ν

  - $\widetilde{\mathtt{pre}}^*(P) \triangleq \nu Z.P \wedge \widetilde{\mathtt{pre}}(Z)$
  - $\mathtt{pre}^*(P) \triangleq \mu Z.P \vee \mathtt{pre}(Z)$
  - $\widetilde{\mathtt{post}}^*(P) \triangleq \nu Z.P \wedge \widetilde{\mathtt{post}}(Z)$
  - $\mathtt{post}^*(P) \triangleq \mu Z.P \vee \mathtt{post}(Z)$

21

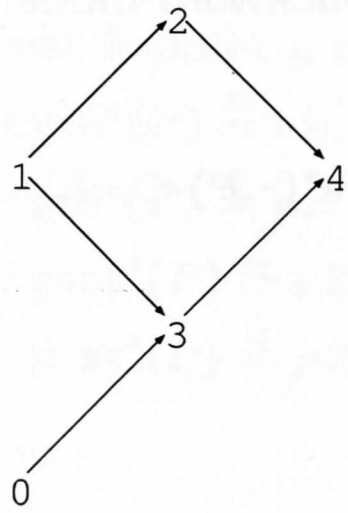
- Equivalence between forward and backward model checking

$$\mathtt{post}^*(I) \subseteq P \Leftrightarrow I \subseteq \widetilde{\mathtt{pre}}^*(P) \Leftrightarrow \mathtt{pre}^*(\neg P) \subseteq \neg I$$

22

State transition systems ($T : X \to \mathcal{P}(A \times X)$)

- $\mathtt{pre}(P) = \{s \in X \mid \exists s' \in X. \exists a \in A. s \xrightarrow{a} s' \wedge s' \in P\}$
- $\widetilde{\mathtt{pre}}(P) = \{s \in X \mid \forall s' \in X. \forall a \in A. s \xrightarrow{a} s' \Rightarrow s' \in P\}$
- $\mathtt{post}(P) = \{s' \in X \mid \exists s \in X. \exists a \in A. s \xrightarrow{a} s' \wedge s \in P\}$
- $\widetilde{\mathtt{post}}(P) = \{s' \in X \mid \forall s \in X. \forall a \in A. s \xrightarrow{a} s' \Rightarrow s \in P\}$



From [Cousot 99]

$\mathtt{pre}(\{3\}) = \{0, 1\}$

$\widetilde{\mathtt{pre}}(\{3\}) = \{0, 4\}$

$\mathtt{post}(\{1\}) = \{2, 3\}$

$\widetilde{\mathtt{post}}(\{1\}) = \{0, 1, 2\}$

23

Galois algebra [von Karger 98]

A complete Boolean algebra $\mathcal{B}$ that has a function $\widetilde{\oplus} : \mathcal{B} \to \mathcal{B}$ preserving all meets (called a "next time" operation).

($\ominus \dashv \widetilde{\oplus}$, $\oplus \triangleq \neg\widetilde{\oplus}\neg$, $\oplus \dashv \widetilde{\ominus}$, $\ominus = \neg\widetilde{\ominus}\neg$ all derivable)

Fact. [Jacobs 99]

For a hidden algebra without a hidden constants, the set $\mathcal{P}(S)$ ($S = S_{h(h \in H)}$) of H-sorted predicates forms a Galois algebra with $\widetilde{\mathtt{pre}}$.

24

(First-Order) Computational Tree Logic (CTL)

CTL formulas

$\phi ::= P \mid \phi \vee \phi \mid \neg\phi \mid \mathbf{EX}\,\phi \mid \mathbf{AX}\,\phi \mid \mathbf{E}\,\phi\,\mathbf{U}\,\phi \mid \mathbf{A}\,\phi\,\mathbf{U}\,\phi.$

(P: state formula)

Abbreviations

$\phi \wedge \psi \triangleq \neg(\neg\phi \vee \neg\psi)$ $\quad \phi \Rightarrow \psi \triangleq \neg\phi \vee \psi$

$\mathbf{EF}\,\phi \triangleq \mathbf{E}\,\top\,\mathbf{U}\,\phi$ $\quad \mathbf{AF}\,\phi \triangleq \mathbf{A}\,\top\,\mathbf{U}\,\phi$

$\mathbf{EG}\,\phi \triangleq \neg\,\mathbf{AF}\,\neg\,\phi$ $\quad \mathbf{AG}\,\phi \triangleq \neg\,\mathbf{EF}\,\neg\,\phi$

25

Interpreting CTL formulas in Galois algebra

$$
\begin{aligned}
p &\triangleq p \text{ (state formula)} \\
\mathtt{EX}\, p &\triangleq \oplus(p) \\
\mathtt{AX}\, p &\triangleq \widetilde{\oplus}(p) \\
\mathtt{EF}\, p &\triangleq \mu z.p \vee \oplus(z) \;= \oplus^*(p) \\
\mathtt{AF}\, p &\triangleq \mu z.p \vee \widetilde{\oplus}(z) \\
\mathtt{AG}\, p &\triangleq \nu z.p \wedge \widetilde{\oplus}(z) \;= \widetilde{\oplus}^*(p) \\
\mathtt{EG}\, p &\triangleq \nu z.p \wedge \oplus(z) \\
\mathtt{E}\, p\, \mathtt{U}\, q &\triangleq \mu z.q \vee (p \wedge \oplus(z)) \\
\mathtt{A}\, p\, \mathtt{U}\, q &\triangleq \mu z.q \vee (p \wedge \widetilde{\oplus}(z))
\end{aligned}
$$

26

Fact. [von Karger 98]

An CTL formula p is valid iff its interpretation in Galois algebra is equal to $\top$.

* Needs a lot of work wrt interpretations of atomic predicates.

27

Behavioral model checking via PigNose

- refutation engine for CafeOBJ
- iterative calculation of fixpoints with $\widetilde{\text{pre}}$ and pre
- first-order predicate logic with equality
- many-sorted resolution / paramodulation (Otter)
- SOS (Set of Support) strategy, demodulation
- also useful for detecting bugs (attribute clash)

28

Lamports's Bakery Algorithm (2-process)

```
N1, N2: Integer := 0;
task body P1 is
begin
  loop
a1:   Non_Critical_Section_1;
b1:   N1 := 1;
c1:   N1 := N2 + 1;
d1:   loop exit when N2 = 0 or N1 <= N2; end loop;
e1:   Critical_Section_1;
f1:   N1 := 0;
  end loop;
end P1;

task body P2 is
...
d2:   loop exit when N1 = 0 or N2 < N1; end loop;
...
```

29

Model checking of mutual exclusion in 5mins

Lockout freedom in similar way

Completely automatic (no abstraction needed)

Easy to extend to n-process case

Comm. protocols, cache coherence, simple control software sytems, ... (no finiteness restriction)

Cryptographic protocol under investigation

30

Future work

- efficient fixpoint calculation
- data type specific decision procedures
  (e.g., Presburger Arithmetic)
- liveness model checking
- abstract interpretation
- real-time hybrid systems

31

形式手法のAからZまで
—Zをどう教えたらよいか—

SIGFM　第5回月例会

2000年7月7日

玉井哲雄
（東京大学大学院情報学環）

形式手法についての個人的な想い

- 世の中は，FMへの「アレルギー反応」と「偏愛」とに二分
- 自分はambivalent（愛憎相半ば）
  - 形式論理のきちんとした修練を経ていないという負い目
    - 野矢茂樹「論理学」
  - 思考の節約に価値
    - 形式と非形式間の自由な飛躍
    - 抽象のレベルの自由な昇降
  - 壷に嵌まった形式化の美しさ

形式化が呼び起こす好悪感

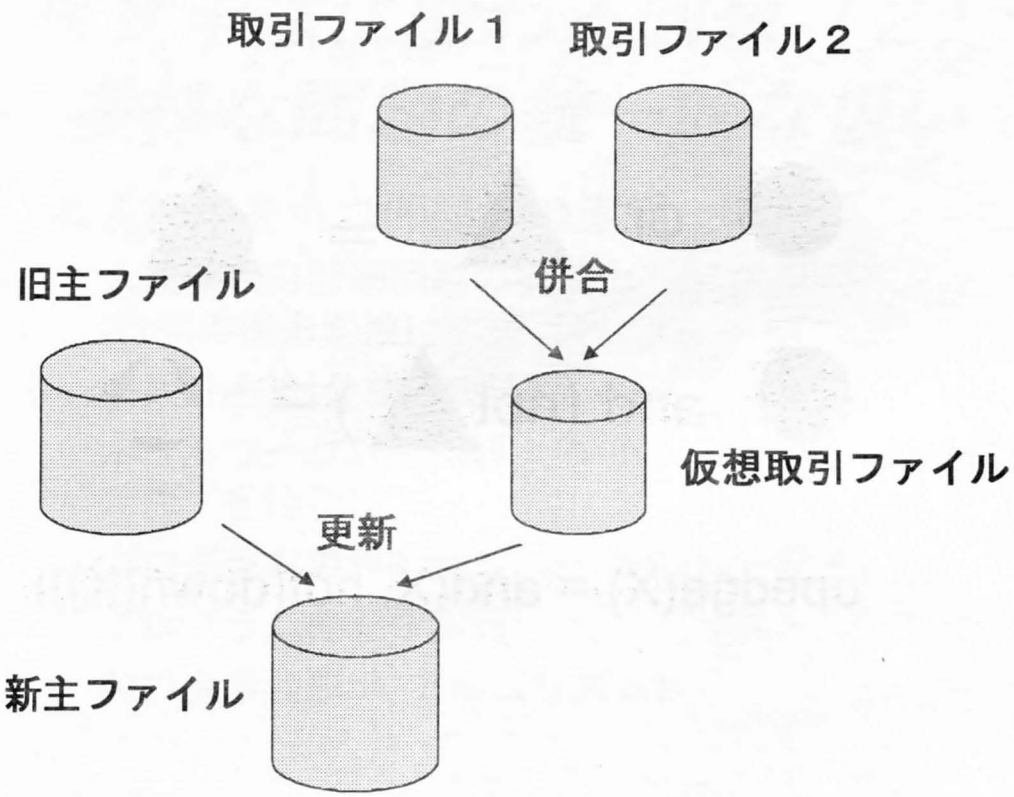
- （とくに日本語で）「形式的」　⇒　中身がない
- 「人の思考は形式的でない」
  - 形式的推論を追うことはできるが面倒
  - モデルに引きつけて理解
- 「抽象化＝形式化ではない」
  - 形式化は抽象化である.
    - 記号化で対象/意味を捨象
  - 非形式的な抽象化もある.

形式手法に関わる回想（1）
ファイル処理プログラムの検証系開発

- 玉井・福永，1977—1980
- 技術的には
  - 半順序に特化した推論機能を持つ定理証明系
  - ループ不変量の自動生成
- より有益だと思われるアイディア
  - 仮想ファイル
    - 同じインターフェースに対する複数の実現
    - 同じ計算対象に対する複数のインターフェース

仮想ファイル

- 背景：n本ファイルの併合，更新，等の処理
  - 例：主ファイルを2本の取引ファイルで更新
    ⇒2本の取引ファイルを併合した仮想ファイルを導入
- ファイルへの操作インタフェース(open, read, eofなど）を定義
- 2本のファイルを実際に併合せず，それへの操作のみを公理を満たすように実現
- 関連する技術
  - ADT
  - ストリーム・プログラミング
  - Javaのimplement句



局面志向プログラミング批判

- Aspect Oriented Programmingの例題：画像処理
  - 自然なモジュール化
    - 白黒画像のand, or, not, up, downなどを用意
    - それらの組み合わせでたとえば縁線抽出などの操作を定義
  - そのままでは速度も記憶容量もむだが多いので，性能を上げるための局面プログラムを書く．
- しかし，仮想ファイルと同じアイディアを使えば，and, or, not, up, down のような基本操作を，実際の画面を作らずに，その特定画素のget/put操作として実現し，モジュール性を保ちながら性能を上げることができる．

upedge(X) = and(X, not(down(X)))

複数のインターフェース(視点)

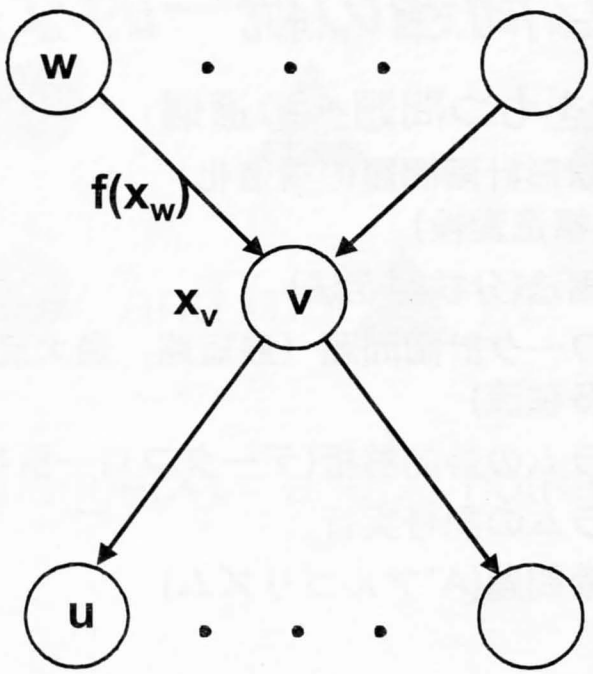
- ファイル＝レコードの並び
  ＝レコード・グループの並び
  ⇒ 「キーの切れ目処理」
- テキスト＝文字の並び
  ＝単語の並び
  ＝行の並び
  ⇒ D. Jacksonのエディタの仕様
- ここでも仮想ファイルの概念が使える.

形式手法に関わる回想（2）多様な問題の統一的な扱い

- 共通構造をもつ問題への遭遇
  - 大規模線形計画問題の構造化(行列の構造変換)
  - 整数計画法(分枝限定法)
  - ネットワーク計画問題（最短路，最大流，最小費用流，多種流)
  - プログラムの静的解析(データフロー解析)
  - プログラムの記号実行
  - AIの探索問題(A*アルゴリズム)

グラフ上の不動点問題として統合化

- グラフと束と束の上の関数空間を用い，不動点方程式として定式化
- アルゴリズムの統一的記述と，正当性の証明
- 山本・萩谷等がHOLで記述，検証
- CafeOBJでも記述．パラメータ・モジュールで個別問題領域に写像．



モデル検査

- 最近はやり (FM99, ICSE2000)
- もとは1980年代初め. Clarke & Emerson'82 など
- モデルが時相論理式を満たすかどうかを自動的に検査
- アルゴリズムはグラフ上の不動点計算
- ハードウェア(論理回路)の検証では日常的に使われている.
- セキュリティ・プロトコルの検査で注目された.

形式手法の意味

- 曖昧性のない厳密な仕様
- 正しさや性質の検証
- "実行"（シミュレーション）
- プログラムの自動合成

本当に実用性があるか？

- 形式手法の利点は広く認識されつつある．
- 実際的な適用例も増えてはきたが，まだ限定的
  - いつも同じ事例 （たとえばCICS）
  - 効果が明らかなのは
    - ハードウェア（論理回路）
    - 通信プロトコル
  - 上の分野ではモデル検査が活躍

ソフトウェアでは

- 従来は高信頼性が要求されるもの
  - 安全性，機密性
- これからは再利用性の高いもの／広い範囲で繰り返し使われるもの
  - ライブラリ，コンポーネント
  - スタンダード
  - 一般性の高い問題と解法

形式的抽象化の方法

- 機能の抽象化
- データの抽象化
- オブジェクトの抽象化
- 時間（順序，並列性）の抽象化

Zの教科書的な導入の仕方

- 状態機械の仕様記述
  - Spiveyの「お誕生日ノート」
- よく使われる表現

Zのくせ

- すべては基本集合の上で作られる.
  - M. Jacksonの批判： 時間的に不変, 互いに素という仮定
- 部分関数の多用
  ← 関数や関係の集合的な定義
- スキーマの引用 ⇒ 簡潔で暗黙な表現 (その意味で分かりにくい)

山崎さんの酒屋問題

ある酒類販売会社の倉庫では，毎日数個のコンテナが搬入されてくる．その内容はビン詰めの酒で，1つのコンテナには10銘柄まで混載できる．扱い銘柄は約200種類ある．倉庫係は，コンテナを受け取りそのまま倉庫に保管し，積荷票を受付係へ手渡す．また受付係からの出庫指示によって内蔵品を出庫することになっている．内蔵品は別のコンテナに詰め替えたり，別の場所に保管することはない．空になったコンテナはすぐに搬出される．

積荷票: コンテナ番号(5桁)
搬入年月，日時
内蔵品名，数量(の繰り返し)

酒屋問題(2)

さて受付係は毎日数十件の出庫依頼を受け，その都度倉庫係へ出庫指示書を出すことになっている．出庫依頼は出庫依頼票または電話によるものとし，1件の依頼では，1銘柄のみに限られている．在庫がないか数量が不足の場合には，その旨依頼者に電話連絡し，同時に在庫不足リストに記入する．また空になるコンテナを倉庫係に知らせることになっている．倉庫内のコンテナ数はできる限り最小にしたいと考えているからである．

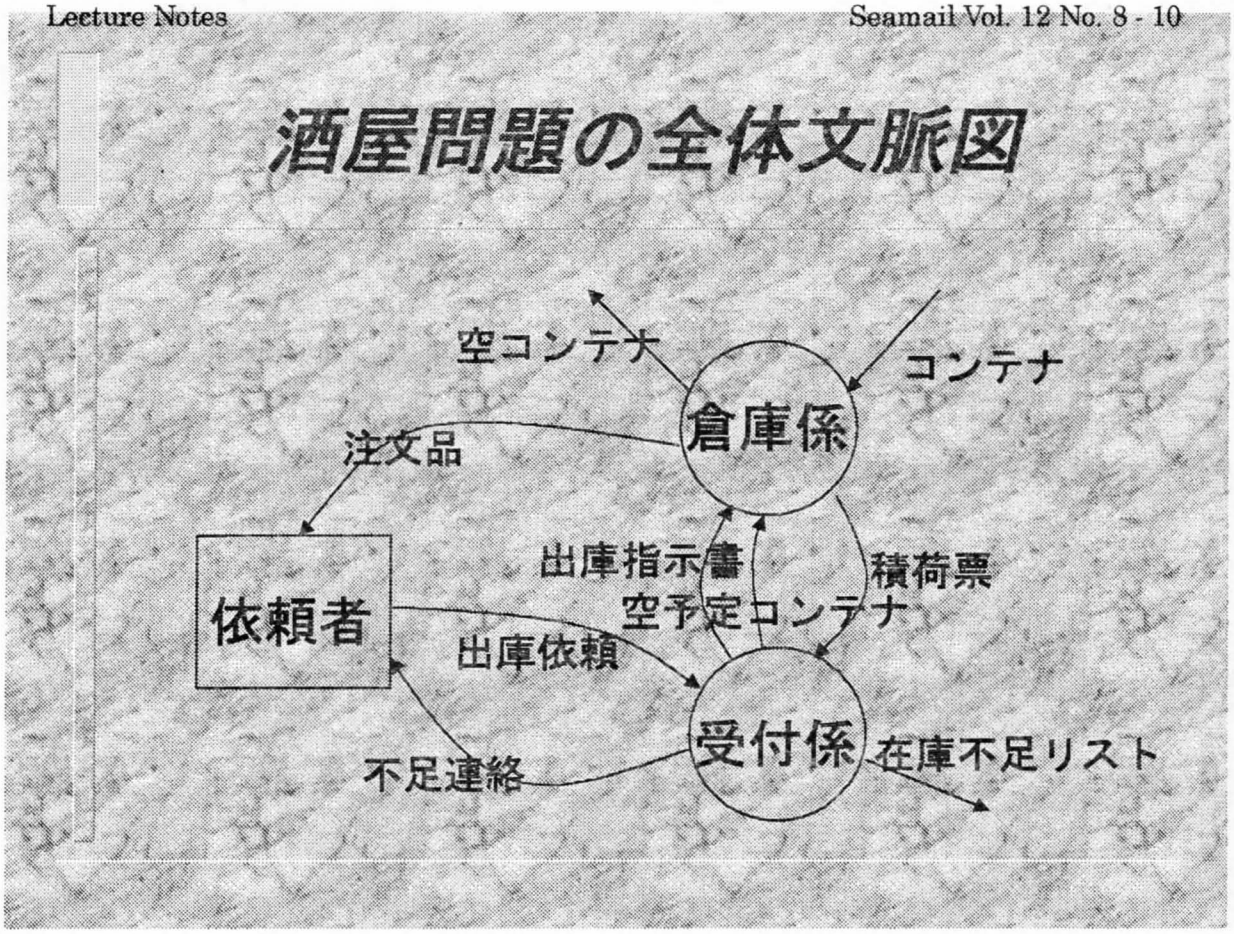
出庫依頼: 品名，数量
　　送り先名

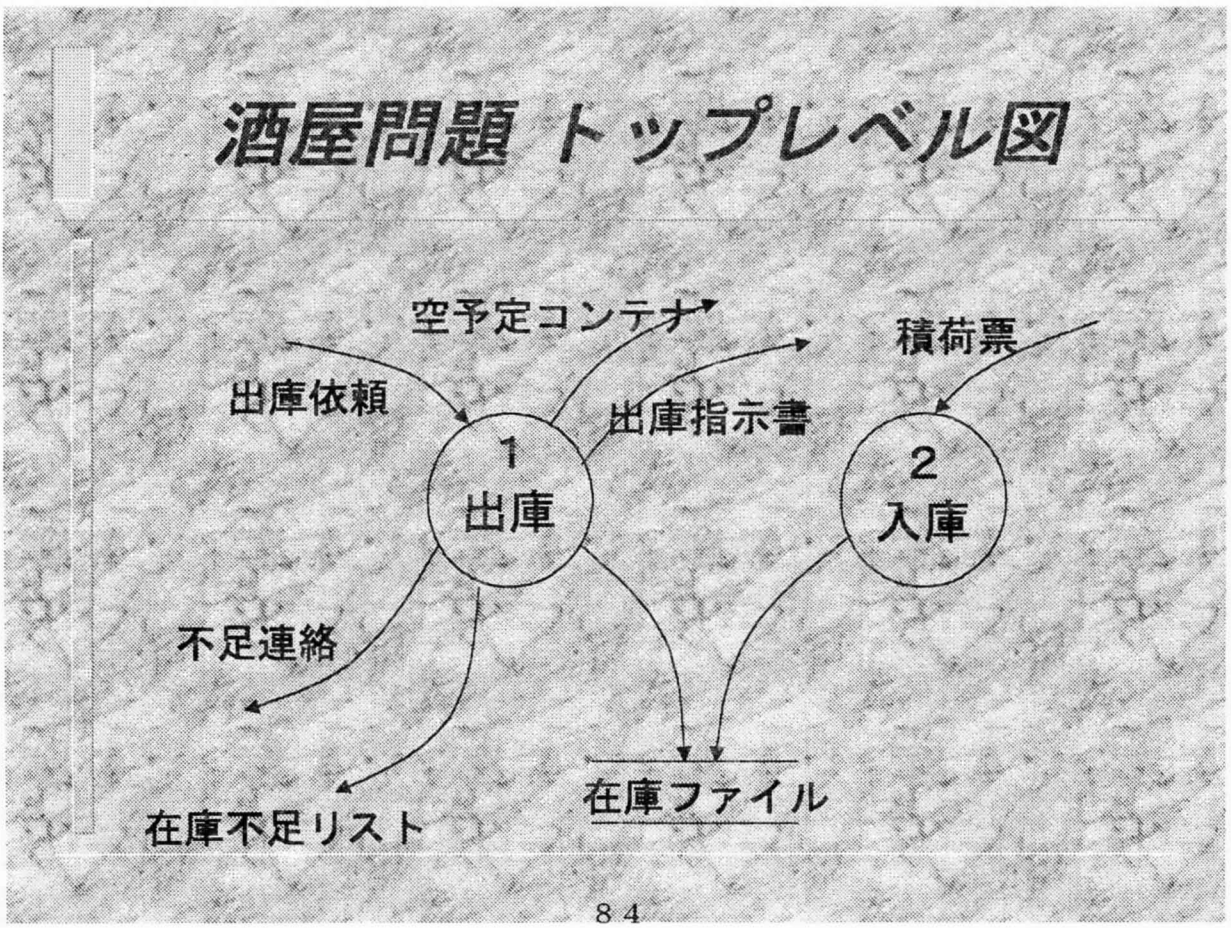
酒屋問題(3)

受付係の仕事(在庫なし連絡，出庫指示書作成および在庫不足リスト作成)のための計算機プログラムを作成せよ．

出庫指示書： 注文番号
　　送り先名
　　コンテナ番号
　　品名，数量　　(の繰り返し)
　　空コンテナ搬出マーク
在庫不足リスト: 送り先名
　　品名，数量

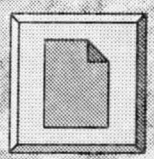
・なお移送や倉庫保管中に酒類の損失は生じない．
・この課題は現実的でない部分もあるので，入力データのエラー処理などは簡略に扱ってよい．


酒屋問題の全体文脈図
空コンテナ
コンテナ
倉庫係
注文品
出庫指示書
積荷票
空予定コンテナ
依頼者
出庫依頼
受付係
在庫不足リスト
不足連絡


酒屋問題 トップレベル図
空予定コンテナ
積荷票
出庫依頼
出庫指示書
1
出庫
2
入庫
不足連絡
在庫ファイル
在庫不足リスト

Zによる酒屋問題の記述

- 高階関数の利用
- 非決定性の利用
- 宣言性の利用
  (入力を出力の関数として表す)



洗練計算

山利治

目的

これは過半数得票問題(Gries-Misra)のCarroll Morganによる解*を紹介し，あわせて洗練計算の雰囲気を味わってもらうことを意図したものである.

*** C. Morgan, Programming from Specifications. Prentice Hall, 1994.**

目的

方法

Lecture Notes Seamail Vol. 12 No. 8 - 10

話の組み立て

1. 洗練計算とは

2. 状態変換子と述語変換子

3. ＲＣ言語

4. 洗練法則

5. 過半数得票選挙

洗練計算 (Refinement Calculus) とは

目的　プログラム作成の算術化

方法　蓋付き命令に仕様文を追加し，プログラム間の洗練関係 $P \sqsubseteq Q$ を定義して，仕様文 P から $P \sqsubseteq P_1 \sqsubseteq \cdots \sqsubseteq P_n = Q$ と実行可能なプログラム Q を法則に従って段階的に導出する．

蓋付き命令 $\cdots$ **Dijkstra の Guarded Commands**

背景

- 構造化プログラミング (段階的洗練)(Naur,Wirth,...)
- プログラムの正当性証明(Floyd,Hoare,...)
- 蓋付き命令と最弱前件によるプログラムの導出 (Dijkstra)

> 書かれたプログラムの検証は楽ではない.
> むしろ始めから正しいプログラムを作るべきである.

- 洗練計算(Back,Morgan,...)
- 形式的方法 B などで洗練計算をそのまま採用(Abrial)

状態変換子と述語変換子

計算 ··· 命令型言語によるプログラムの計算は命令が及ぼす状態(プログラム変数がとる値)の変化の累積効果である.

状態変換子 (state transformer)··· プログラムを状態から状態への(部分多重)関数とみなしたときにいう.

述語変換子(predicate transformer)··· 状態変換子に対してその逆関数をいう. 但し状態の冪集合からそれへの関数である.

ここでは状態集合$S = \{\sigma \in \Sigma | p\}$を述語$p$とみなす.

参考

任意の多重関数$f : X \to Y$に対してつぎのような関数$f_P : \mathbb{P}(Y) \to \mathbb{P}(X) : S \mapsto \{x \in X | f(x) \subseteq S\}$が定義できる.
X,Yをプログラムの変数から値への写像全体すなわち状態とするとき,上のようなf, f_Pを状態変換子，述語変換子という.

状態変換子としてのプログラム

$$P \equiv \textbf{if } x \geq y \to z := x \ \square \ x \leq y \to z := y \textbf{ fi}$$

$$P : \begin{pmatrix} x \mapsto 1 \\ y \mapsto 2 \\ z \mapsto 3 \end{pmatrix} \mapsto \begin{pmatrix} x \mapsto 1 \\ y \mapsto 2 \\ z \mapsto 2 \end{pmatrix}$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

最弱前件

プログラムを述語変換子とみる:

$p\ P\ q$ ··· プログラムPと状態上の述語p, qに対してpが満たされる状態
プログラムPを実行すると必ず停止してそのときの状態はqを満たす.

$[P]q$ ··· つぎのように定義した述語でPのqに対する最弱前件**(weakest pr condition)**という.

$$p\ P\ q \Leftrightarrow (p \Rightarrow [P]q)$$

つまり述語$[P]q$はPを,そこから実行すると必ず停止して述語qが成立す
ようなもののうち一番弱い条件を意味する.

述語変換子としてのプログラム

$$P \equiv \textbf{if}\ x \geq y \rightarrow z := x \ \square\ x \leq y \rightarrow z := y\ \textbf{fi}$$

$$[P](\clubsuit) : (z = y + 1) \mapsto (x = y + 1)$$

つまり:

$$[P](z = y + 1) = (x = y + 1)$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

RC言語

$$c ::= \mathbf{skip} \mid [\, p_1, p_2 \,] \mid x := e \mid c_1; c_2 \mid \mathbf{if}\ gc\ \mathbf{fi} \mid \mathbf{do}\ gc\ \mathbf{od}$$
$$gc ::= b \to c \mid gc_1 \Box\ gc_2$$

$x \in$ 変　数
$a \in$ 算術式
$b \in$ 論理式
$p \in$ 述　語

RC言語 $\cdots$ Refinement Calculus言語

RC言語の最弱前件意味論

$$[\mathbf{skip}\,]r = r$$
$$[p,q\,]r = p \wedge (\forall x \bullet q \Rightarrow r)[v_0 \backslash v]$$
$$[x := e\,]r = r[x \backslash e]$$
$$[c_1; c_2\,]r = [\, c_1\,]([\, c_2\,]r)$$
$$[\mathbf{if}\ b_1 \to c_1 \Box\ b_2 \to c_2\ \mathbf{fi}\,]r = (b_1 \vee b_2) \wedge (b_1 \Rightarrow [\, c_1\,]r \wedge b_2 \Rightarrow [\, c_2\,]r)$$
$$[\mathbf{do}\ b_1 \to c_1 \Box\ b_2 \to c_2\ \mathbf{od}\,]r = \exists k : \mathbb{N} \bullet H_k(r)$$
$$\textit{where}\ H_0(r) = \neg(b_1 \vee b_2) \wedge r,$$
$$H_k(r) = H_0(r) \vee [\ \mathbf{if}\ b_1 \to c_1 \Box b_2 \to c_2\ \mathbf{fi}\ ](H_{k-1}(r))$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

仕様文(Specification Statement)

$x, y, ... : [\, p, q \,]$ p, qは変数$x, y, ...$を含む述語で仕様文$[\ p, q\ ]$はpが成立する状態から実行すると$x, y, ...$だけを変更してqが成立する状態で停止するプログラムを指す．これは洗練しなければ実行不能である．

略記法

$$
\begin{aligned}
\{p\} &= [\, p, true \,] \\
[\, p \,] &= [\, true, p \,] \\
[\, p, i, q \,] &= [\, p \wedge i, q \wedge i \,]
\end{aligned}
$$

特別な例

$$
\begin{aligned}
[\, true, true \,] &= \textbf{skip} \\
[\, true, false \,] &= \textbf{magic} \\
[\, false, true \,] &= \textbf{abort} \\
[\, false, false \,] &= \textbf{abort}
\end{aligned}
$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

RC言語の拡張

$[\![\mathbf{var\ x\ \bullet\ c}]\!]$	局所変数を伴なう局所区画の導入
$[\![\mathbf{con\ x\ \bullet\ c}]\!]$	局所定数を伴なう局所区画の導入
$x, y := e, f$	多重代入文

定義

- $[\,[p, q]\,]r \;=\; p \wedge (\forall X \;\bullet q \Rightarrow r)$ **(**Xは今宣言している変数リスト**)**

- $[\,Y : [p, q]\,]r \;=\; p \wedge (\forall X \;\bullet q \Rightarrow r)[X_0 \backslash X]$ **(**Yは今宣言している変数リストXの一部分でX_0はXの初期値**)**

Lecture Notes Seamail Vol. 12 No. 8 - 10

定理

⊙ $p \wedge X = X_0 \Rightarrow [\ Z : [p, q]\ ](q \wedge Y = Y_0)$ (Y, Z は X の分割)

洗練 (Refinement)

$P \sqsubseteq Q$ (Q は P の洗練) $\longleftrightarrow \forall\, p \bullet [P]p \Rightarrow [Q]p$

$$P = \textbf{if}\ 2|x \rightarrow x := x + 2 \ \square\ 3|x \rightarrow x := x + 3\ \textbf{fi}$$
$$Q = \textbf{if}\ 2|x \rightarrow x := x + 2 \ \square\ \neg(2|x) \rightarrow x := x + 3\ \textbf{fi}$$

ここでは $P \sqsubseteq Q$ になる. $\because$

$[P]r = (2|x \vee 3|x) \wedge (2|x \Rightarrow r[x \backslash x + 2]) \wedge (3|x \Rightarrow r[x \backslash x + 3])$

$[Q]r = (2|x \Rightarrow r[x \backslash x + 2]) \wedge (\neg(2|x) \Rightarrow r[x \backslash x + 3])$

$\therefore \forall\, r \bullet [P]r \Longrightarrow [Q]r$

補足

つぎが成り立つ:

$\{p\} \sqsubseteq \mathbf{skip}$

$\forall p, q \bullet x : [\, p, q \,] \sqsubseteq \mathbf{magic}$

magicはしかし洗練できない.

定理

$p \wedge X = X_0 \Rightarrow [P](q \wedge Y = Y_0)$ ならば

$[\, Z : [p, q] \,] \sqsubseteq P$ **(Y, Z は X の分割)**

Lecture Notes

Seamail Vol. 12 No. 8 - 10

洗練法則

(1) $\dfrac{Y\colon [\, p,r \,]}{Y\colon [\, q,r \,]} \quad (p \Rightarrow q)$

(2) $\dfrac{Y\colon [\, p,q \,]}{Y\colon [\, p,r \,]} \quad (p \Rightarrow (\forall X \bullet r \Rightarrow q)[X_0 \backslash X])$

(3) $\dfrac{Y,x\colon [\, p,q \,]}{Y\colon [\, p,q \,]}$

(4) $\dfrac{Y\colon [\, p,q \,]}{|[\ \mathbf{var}\ x;\ Y,x\colon [\, p,q \,]\ ]|} \quad (x$は新変数$)$

(5) $\dfrac{Y\colon [\, false,p \,]}{\mathbf{abort}}$

例

(1) $x : [\, x > 0 \ ,\ x = x_0 - 1 \,] \sqsubseteq x : [\, x \geq 0 \ ,\ x - x_0 - 1 \,]$

(2) $x : [\, x \geq 0 \,] \sqsubseteq x : [\, x > 0 \,]$

(3) $x, y : [\, x = y_0 \,] \sqsubseteq x : [\, x = y_0 \,]$

(4) $x : [\, x = n! \,] \sqsubseteq |[\ \mathbf{var}\ i;\ x, i : [\, x = n! \,]\ ]|$

(5) $x : [\, x < 0 \wedge x > 0, p(x) \,] \sqsubseteq x := 7$

Lecture Notes Seamail Vol. 12 No. 8 - 10

洗練法則(つづき)

(6) $\dfrac{Y\!:[\,[p[X_0\backslash X,q]\,]}{\textbf{skip}}$

(7) $\dfrac{Y\!:[\,p[X_0,Y\backslash X,E],p\,]}{Y:=E}$

(8) $\dfrac{Y\!:[\,p,\,q\,]}{Y\!:[\,p,[P]q\,]\;;\;P}(Y:[\,true\,]\sqsubseteq P)$

(9) $\dfrac{Y\!:[\,p,\,q\,]}{Y\!:[\,p,m\,]\;;\;Y\!:[\,m,\,q\,]}$ $(m,q$は自由なX_0を含まない)

例

(6) $x,y:[\,x=y,\;x=y_0\,]\sqsubseteq \textbf{skip}$

(7) $x:[\,x=x_0+y\,]\sqsubseteq x:=x+y$

(8) $x,y:[\,x=y+1\,]\sqsubseteq x,y:[\,x=2\,];\;y:=1$
$Y:[\,true\,]\sqsubseteq P$ PはYだけを変える

(9) $x:[\,x=y+1\,]\sqsubseteq x:[\,x=y\,];\;x:[\,x=y,\;x=y+1\,]$

Lecture Notes Seamail Vol. 12 No. 8 - 10

洗練法則(つづき)と例

(10) $\dfrac{Y\colon [\, p \wedge (\bigvee i \bullet b_i),\ q \,]}{\textbf{if } (\Box i \bullet b_i \to Y\colon [\, p \wedge b_i,\ q \,]) \textbf{ fi}}$

$y : \ [x = 0 \vee x = 1,\ x + y = 1]$

$\sqsubseteq$ **if** $x = 0 \ \to \ y : \ [\, x = 0,\ x + y = 1 \,]$
$\Box \ x = 1 \ \to \ y : \ [\, x = 1,\ x + y = 1 \,]$
fi

$\sqsubseteq$ **if** $x = 0 \ \to \ y := 1$
$\Box \ x = 1 \ \to \ y := 0$
fi

洗練法則(つづき)

(10) $\dfrac{Y\colon [\, true,\ inv,\ \neg(\bigvee i \bullet b_i) \,]}{\textbf{do } (\Box\ i \bullet b_i \to Y\colon [\, b_i,\ inv,\ 0 \leq var < var_0 \,]) \textbf{ od}}$

反復文の導入

自然数a, bの最大公約数cを求めよ.

a,bの最大公約数を(a, b)と略記する.
求めるプログラムは:

$$
\begin{array}{l}
[\![\ \mathbf{con}\ a, b : \mathbb{N} \bullet \\
\quad c : [\ a > 0 \wedge b > 0,\ c = (a, b)\ ] \\
]\!]
\end{array}
$$

反復文の導入(つづき)

注意・・・つぎが成立する:

$$
\begin{array}{rcl}
(x, y) & = & (y, x) \\
(x, y) & = & (x, y - x) \\
(x, y) & = & (x - y, y) \\
(x, x) & = & x
\end{array}
$$

$$inv \equiv x > 0 \ \wedge \ y > 0 \ \wedge \ (x, y) \ = \ (a, b)$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

反復文の導入(つづき)

$$
\begin{array}{ll}
\sqsubseteq & |[\mathbf{var}\ c, x, y : \mathbb{N} \bullet \\
& \quad x, y := a, b\ ; \\
& \quad x, y : [\ true,\ inv,\ \neg(x > y \vee x < y)\ ]\ ; \qquad \heartsuit \\
& \quad x, c : [\ x = (a, b),\ c = (a, b)\ ] \qquad \spadesuit \\
&]|
\end{array}
$$

反復文の導入(つづき)

$\heartsuit$

$$
\begin{array}{lll}
\sqsubseteq & \mathbf{do} & x > y \ \rightarrow\ x, y : [\ x > y,\ inv,\ 2 \leq x + y < a + b\ ] \qquad \clubsuit \\
& \square & y > x \ \rightarrow\ x, y : [\ x < y,\ inv,\ 2 \leq x + y < a + b\ ] \qquad \diamondsuit \\
& \mathbf{od} &
\end{array}
$$

$$
\begin{array}{lll}
\clubsuit & \sqsubseteq & x := x - y \\
\diamondsuit & \sqsubseteq & y := y - x \\
\spadesuit & \sqsubseteq & c := x
\end{array}
$$

反復文の導入(つづき)

結局プログラムはつぎのようになる:

$$
\begin{array}{l}
[\![\ \mathbf{con}\ a, b : \mathbb{N}\ ; \\
\quad \mathbf{var}\ x, y, c : \mathbb{N}\ ; \\
\quad x, y\ :=\ a, b\ ; \\
\quad \mathbf{do}\ x > y\ \rightarrow\ x\ :=\ x - y \\
\quad \Box\ \ x < y\ \rightarrow\ y\ :=\ y - x \\
\quad \mathbf{od}\ ; \\
\quad c\ :=\ x \\
\ ]\!]
\end{array}
$$

過半数得票選挙

ある会の会長を選ぶ選挙がある.
会長は会員全員の投票で過半数得票者が選ばれる.
会長が誰かを知るプログラムを書け.

$$
\begin{aligned}
\mathbf{sm}\ x\ a\ &\equiv\ a.x\ >\ (\#a) \div 2 \\
\mathbf{em}\ a\ &\equiv\ (\exists x : T\ \bullet\ \mathbf{sm}\ x\ a) \\
\mathbf{cm}\ x\ a\ &\equiv\ \mathbf{em}\ a\ \Rightarrow \mathbf{sm}\ x\ a
\end{aligned}
$$

★ $a.x$ は有限列aのなかに含まれているxの数.
★ $\#a$ は有限列aの長さ.

Lecture Notes Seamail Vol. 12 No. 8 - 10

プログラム仕様

$$\mathbf{var}\ a:\ \mathbf{seq_N}\ T;\ x:\ T \bullet$$
$$x:\ [\ \mathbf{cm}\ x\ a\ ]$$

★ $\mathbf{seq_N}$ は長さNの有限列型.

★ $a \uparrow k$ は有限列aの長さkの先頭部分列.

★ $a[k]$は有限列aのk番目の要素但し0番から始める.

★ $a[i \to j]$は$a[i], a[i+1], \cdots a[j-1]$がつくる$a$の長さ$j-i$の部分列.

洗練

$\sqsubseteq$ $\mathbf{var}\ i:\ \mathbb{N} \bullet$
$i\ :=\ 0\ ;$
$\mathbf{do}\ \ i \neq N \to$
$\quad x:\ [\ \mathbf{cm}\ x\ a \uparrow i\ ,\mathbf{cm}\ x\ a \uparrow (i+1)\ ]\ ;$
$\quad i\ :=\ i+1$
$\mathbf{od}$

洗練つづき

$$
\begin{array}{ll}
\sqsubseteq & \textbf{if}\ \neg\mathbf{em}\ a\uparrow i \to\ x : [\ \neg\mathbf{em}\ a\uparrow i\ ,\ \mathbf{cm}\ x\ a\uparrow(i+1)\ ] \quad \clubsuit \\
& \square\ \mathbf{sm}\ x\ a\uparrow i \to\ x : [\ \mathbf{sm}\ x\ a\uparrow i\ ,\ \mathbf{cm}\ x\ a\uparrow(i+1)\ ] \quad \diamondsuit \\
& \textbf{fi}
\end{array}
$$

$\clubsuit \quad \sqsubseteq \quad x := a[i]$

$\diamondsuit \quad \sqsubseteq \quad \mathbf{skip}$

N.B. $\mathbf{cm} \Leftrightarrow \neg\mathbf{em} \vee \mathbf{sm}$

$\therefore$ 選択文が導入できた.

洗練つづき

補助変数$c = a\uparrow i.x$を導入してプログラムを簡単にしたい.

Lecture Notes Seamail Vol. 12 No. 8 - 10

洗練つづき

$$
\begin{array}{l}
[\![\ \mathbf{var}\ i\ :\ \mathbb{N} \bullet \\
\quad i\ :=\ 0\ ; \\
\quad \mathbf{do}\ i \neq N\ \rightarrow \\
\qquad \mathbf{if}\ \ \neg \mathbf{em}\ a \uparrow i\ \rightarrow\ x\ :=\ a[i] \\
\qquad \Box\ \ \mathbf{sm}\ x\ a \uparrow i\ \rightarrow\ \mathbf{skip} \\
\qquad \mathbf{fi}\ ; \\
\qquad i\ :=\ i+1 \\
\quad \mathbf{od} \\
]\!]
\end{array}
$$

$$
\begin{array}{l}
[\![\ \mathbf{var}\ i, c\ :\ \mathbb{N} \\
\quad \mathbf{initially}\ c\ =\ a \uparrow i.x \bullet \\
\quad i, c\ :=\ 0, 0\ ; \\
\quad \mathbf{do}\ i \neq N\ \rightarrow \\
\qquad \mathbf{if}\ \ c \leq i \div 2\ \rightarrow\ x, c\ :=\ a[i], (a \uparrow i).a[i] \\
\qquad \Box\ \ c > i \div 2\ \rightarrow\ \mathbf{skip} \\
\qquad \mathbf{fi}\ ; \\
\qquad \mathbf{if}\ a[i] = x\ \rightarrow\ i, c\ :=\ i+1, c+1 \\
\qquad \Box\ a[i] \neq x\ \rightarrow\ i\ :=\ i+1 \\
\qquad \mathbf{fi} \\
\quad \mathbf{od} \\
]\!]
\end{array}
$$

洗練つづき

$\sqsubseteq\ x := a[i]\ ;\ c := a \uparrow i.x$

$\sqsubseteq\ inv\ c = a \uparrow j.x \wedge j \leq i;\ \mathbf{var}\ j\ :\ \mathbb{N} \bullet$

$\quad j, c := 0, 0\ ;$

$\quad \mathbf{do}\ j \neq i\ \rightarrow$

$\qquad \mathbf{if}\ x = a[j]\ \rightarrow\ c, j := c+1, j+1$

$\qquad \Box\ x \neq a[j]\ \rightarrow\ j := j+1$

$\qquad \mathbf{fi}$

$\quad \mathbf{od}$

これは残念ながら N の **2** 乗に比例する計算量をもつプログラムである．改良したい．

準備:補題 1

補題 1	$\forall\, a, b : \mathbf{seq}\ T \bullet$ $\mathbf{sm}\ x\ (a++b)\ \Rightarrow \mathbf{sm}\ x\ a\ \vee \mathbf{sm}\ x\ b$

$\because$ **(対偶を証明)**

$\neg\,(\mathbf{sm}\ x\ a \vee \mathbf{sm}\ x\ b) \equiv (a.x \leq \#a \div 2) \wedge (b.x \leq \#b \div 2)$

$\therefore\ a.x + b.x = (a++b).x \leq \#a \div 2 + \#b \div 2 \leq \#(a++b) \div 2$

$i.e.\quad \neg\ \mathbf{sm}\ x\ (a++b)$

Lecture Notes Seamail Vol. 12 No. 8 - 10

準備:補題 2

補題 2 $\forall\, a, b : \mathbf{seq}\ T \bullet$
$\neg\, \mathbf{em}\ a \ \wedge\ \mathbf{cm}\ x\ b \ \Rightarrow\ \mathbf{cm}\ x\ (a + +b)$

注意

$\neg\, \mathbf{em}\ a \ \Rightarrow\ \neg\, \mathbf{sm}\ x\ a$

$\mathbf{cm}\ x\ b \ \equiv\ \neg\, \mathbf{em}\ b \ \vee\ \exists\, x \bullet\ \mathbf{sm}\ x\ b$

1) $\neg\, \mathbf{em}\ b \ \Rightarrow\ \neg\, \mathbf{em}\ (a + +b)$ **(補題 1 と** $\neg\, \mathbf{em}\ a$ **から)**
$\Rightarrow\ \mathbf{cm}\ x\ (a + +b)$

2) $\exists\, x \bullet\ \mathbf{sm}\ x\ b \ \Rightarrow\ \forall\, y \neq x \bullet \neg\, \mathbf{sm}\ y\ b$
$\Rightarrow\ \forall\, y \neq x \bullet \neg\, \mathbf{sm}\ y\ (a + +b)$ **(補題 1)**
$\Rightarrow\ \mathbf{cm}\ x\ (a + +b)$

改良版

$x\ :\ [\mathbf{cm}\ x\ a]$

$\sqsubseteq$ $\mathbf{var}\ j\ :\ \mathbb{N} \bullet$
$x, j\ :\ [\neg \mathbf{em}\ a[0 \to j] \wedge \mathbf{cm}\ x\ a[j \to N]]$

$\sqsubseteq$ $I \ \triangleq\ \neg \mathbf{em}\ a[0 \to j] \wedge \mathbf{cm}\ x\ a[j \to i]$
$\mathbf{var}\ i\ :\ \mathbb{N} \bullet$
$x, j, i\ :\ [I \wedge i = N]$

改良版

$\sqsubseteq$ "$inv \quad I \wedge j \leq i \leq N$"
$i, j := 0, 0$;
do $i \neq N \rightarrow$
  if $\neg$**em** $a[0 \rightarrow j] \wedge \neg$**em** $a[j \rightarrow i] \rightarrow$
    $x, j : [\neg$**em** $a[0 \rightarrow j] \wedge \neg$**em** $a[j \rightarrow i], I[i \backslash i+1]]$ ♣
  $\Box \neg$**em** $a[0 \rightarrow j] \wedge$ **sm** $x\ a[j \rightarrow i] \rightarrow$
    $x, j : [\neg$**em** $a[0 \rightarrow j] \wedge$ **sm** $x\ a[j \rightarrow i], I[i \backslash i+1]]$ ♢
  fi;
  $i := i + 1$
od

改良版

♣ $\sqsubseteq$ $j, x := i, a[i]$

♢ $\sqsubseteq$ **skip**

Lecture Notes Seamail Vol. 12 No. 8 - 10

変形

ここでも変数$c = a[j \to i].x$を導入する.
$g_1 \vee g_2$が成立する状態で$h_1 \vee h_2$が成立し,
そのときまた$h_i \Rightarrow g_i$も成立するなら, つぎがいえる.

$$\textbf{if } g_1 \to prog_1 \,\square\, g_2 \to prog_2 \textbf{ fi}$$
$$\sqsubseteq \quad \textbf{if } h_1 \to prog_1 \,\square\, h_2 \to prog_2 \textbf{ fi}$$

そこでつぎのプログラムがえられる.

線形コード

$$
\begin{array}{l}
[\![\quad \textbf{var } i, j, c : \mathbb{N} \bullet \\
\\
\quad i, j, c := 0, 0, 0 \,; \\
\quad \textbf{do} \quad i \neq N \quad \to \\
\qquad \textbf{if} \quad c \leq (i-j) \div 2 \quad \to \quad i, x, c := i, a[i], 0 \\
\qquad \square \quad c > (i-j) \div 2 \quad \to \quad \textbf{skip} \\
\qquad \textbf{fi} \,; \\
\qquad \textbf{if} \quad a[i] = x \quad \to \quad i, c := i+1, c+1 \\
\qquad \square \quad a[i] \neq x \quad \to \quad i := i+1 \\
\qquad \textbf{fi} \\
\quad \textbf{od} \\
]\!]
\end{array}
$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

線形コード (変形1)

ここでさらに変数$d : \mathbb{Z} = 2c - (i - j)$を導入し，$c, j$を除去する．
選択文の蓋は$d \leq 0$と$d > 0$になりc, jを除く前のプログラムが変形１で，c, jを除いたプログラムが変形２である．

線形コード (変形1)

$$
\begin{array}{l}
[\![\ \ \textbf{var}\ i, j, c\ :\ \mathbb{N}\ ;\ d\ :\ \mathbb{Z} \bullet \\
\\
\quad i, j, c, d\ :=\ 0, 0, 0, 0\ ; \\
\quad \textbf{do}\quad i \neq N\ \rightarrow \\
\qquad \textbf{if}\quad d \leq 0\ \rightarrow\ j, x, c, d\ :=\ i, a[i], 0, 0 \\
\qquad \Box\quad d > 0\ \rightarrow\ \textbf{skip} \\
\qquad \textbf{fi}\ ; \\
\qquad \textbf{if}\quad a[i] = x\ \rightarrow\ i, c, d\ :=\ i + 1, c + 1, d + 1 \\
\qquad \Box\quad a[i] \neq x\ \rightarrow\ i, d\ :=\ i + 1, d - 1 \\
\qquad \textbf{fi} \\
\quad \textbf{od} \\
]\!]
\end{array}
$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

線形コード (変形2)

$$
\begin{array}{l}
[\![\quad \mathbf{var}\ i\ :\ \mathbb{N}\ ;\ d\ :\ \mathbb{Z} \bullet \\
\\
\quad i, d\ :=\ 0, 0\ ; \\
\quad \mathbf{do} \quad i \neq N\ \rightarrow \\
\qquad \mathbf{if} \quad d \leq 0\ \rightarrow\ x, d\ :=\ a[i], 0 \\
\qquad \Box \quad d > 0\ \rightarrow\ \mathbf{skip} \\
\qquad \mathbf{fi}\ ; \\
\qquad \mathbf{if} \quad a[i] = x\ \rightarrow\ i, d\ :=\ i+1, d+1 \\
\qquad \Box \quad a[i] \neq x\ \rightarrow\ i, d\ :=\ i+1, d-1 \\
\qquad \mathbf{fi} \\
\quad \mathbf{od} \\
]\!]
\end{array}
$$

選択文の変形法則

変形2の2つの選択文は独立ではない．$d \leq 0$の分枝から$a[i] \neq x$のそれへは続かない．選択文のつぎの変形法則を注意する：

$$
\begin{array}{l}
\mathbf{if}\ (\Box i \cdot b_i\ \rightarrow prog_i)\ \mathbf{fi}\ ; prog \\
\quad = \mathbf{if}\ (\Box i \cdot b_i\ \rightarrow prog_i ;\ prog)\ \mathbf{fi}
\end{array}
$$

$$
\begin{array}{l}
x\ :=\ e\ ;\ \mathbf{if}\ (\Box i \cdot b_i\ \rightarrow prog_i)\ \mathbf{fi} \\
\quad = \mathbf{if}\ (\Box i \cdot b_i[x \backslash e]\ \rightarrow x\ :=\ e ;\ prog_i)\ \mathbf{fi}
\end{array}
$$

この法則を変形2に適用したものが変形3である．

線形コード (変形3)

$$
\begin{array}{l}
\textbf{if} \quad d \leq 0 \ \rightarrow \\
\quad x, d \ := \ a[i], 0 \, ; \\
\quad \textbf{if} \quad a[i] = x \ \rightarrow \ d \ := \ d+1 \\
\quad \Box \quad a[i] \neq x \ \rightarrow \ d \ := \ d-1 \\
\quad \textbf{fi} \\
\Box \quad d > 0 \ \rightarrow \\
\quad \textbf{if} \quad a[i] = x \ \rightarrow \ d \ := \ d+1 \\
\quad \Box \quad a[i] \neq x \ \rightarrow \ d \ := \ d-1 \\
\quad \textbf{fi} \\
\textbf{fi} \, ; \\
i \ := \ i+1
\end{array}
$$

選択文の変形法則

変形3の蓋を見なおしたものが変形4であり, $true, false$ の蓋をもつ選択文の変形法則によって書きなおしたものが変形5である.

$$
\begin{array}{l}
\textbf{if} \ (\Box \, i \cdot b_i \ \rightarrow \ prog_i \, \Box \, false \ \rightarrow \ prog) \ \textbf{fi} \\
\quad = \ \textbf{if} \ (\Box \, i \cdot b_i \ \rightarrow \ prog_i) \ \textbf{fi}
\end{array}
$$

$$
\textbf{if} \ true \ \rightarrow \ prog \ \textbf{fi} \ = \ prog
$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

線形コード (変形4)

if $d \leq 0 \rightarrow$
  if $true \rightarrow x, d := a[i], 0 \,;\, d := d+1$
  □ $false \rightarrow x, d := a[i], 0 \,;\, d := d-1$
  fi
□ $d > 0 \rightarrow$
  if $a[i] = x \rightarrow d := d+1$
  □ $a[i] \neq x \rightarrow d := d-1$
  fi
fi

線形コード (変形5)

if $d \leq 0 \rightarrow x, d := a[i], 1$
□ $d > 0 \rightarrow$
  if $a[i] = x \rightarrow d := d+1$
  □ $a[i] \neq x \rightarrow d := d-1$
  fi
fi

選択文の変形法則

変形5の選択文の入れ子を展開して簡単にしたい.
つぎはそのための変形法則である.

$$\begin{aligned}&\textbf{if}\,(\Box\, i\cdot b_i \rightarrow \textbf{if}\,(\Box\, i\cdot b_i \rightarrow prog_{ij})\,\textbf{fi})\textbf{fi}\\ &\quad = \textbf{if}\,(\Box\, i,j\cdot b_i \wedge h_j \rightarrow prog_{ij})\,\textbf{fi}\end{aligned}$$

これを適用したものが変形**6**である.

線形コード (変形6)

$$\begin{aligned}&\textbf{if}\quad d \leq 0 \rightarrow x,d := a[i],1\\ &\Box\quad d>0 \wedge a[i]=x \rightarrow d := d+1\\ &\Box\quad d>o \wedge a[i]\neq x \rightarrow d := d-1\\ &\textbf{fi}\end{aligned}$$

Lecture Notes Seamail Vol. 12 No. 8 - 10

変形

変形6をみると$d \geq 0$は初期条件が満たし代入文の実行もそれを保存するから不変式である．蓋は$d = 0$と$d \neq 0$としていい．そして$-$を$\ominus$に変えdの型を$\mathbb{N}$にする．そうしたものが変形7である．

N.B. $a - b \geq 0$なら$a \ominus b = a - b$

線形コード (変形7)

if $d = 0 \rightarrow x, d := a[i], 1$
□ $d \neq 0 \wedge a[i] = x \rightarrow d := d + 1$
□ $d \neq 0 \wedge a[i] \neq x \rightarrow d := d \ominus 1$
fi

Lecture Notes Seamail Vol. 12 No. 8 - 10

変形

プログラムを簡単にするために不変式を考えた．蓋を簡単にするために蓋と初期化についての変形法則をつぎに示す．変形**7**にこれを適用したものが変形**8**である．

不変式が成立する状態で蓋bと蓋b'が同値であればb'をbの代わりに使用していい．

$\exists t T \cdot I$と初期条件Iを変更して補助変数を除去できる．

線形コード (変形8)

if $d = 0 \rightarrow x, d := a[i], d + 1$
□ $d \neq 0 \wedge a[i] = x \rightarrow x, d := a[i], d + 1$
□ $d \neq 0 \wedge a[i] \neq x \rightarrow d := d \ominus 1$
fi

Lecture Notes Seamail Vol. 12 No. 8 - 10

変形

変形8では選択文の分枝に共通文をもつものがある．それをまとめる変形法則をつぎに示す．変形8にこれを適用してまとめたものが変形9である．

$$
\begin{array}{l}
\textbf{if}\ (\Box i \cdot b_i \rightarrow prog_i) \\
\quad \Box\ b1 \rightarrow prog \\
\quad \Box\ b2 \rightarrow prog \\
\quad \textbf{fi} \\
= \ \textbf{if}\ (\Box i \cdot b_i \rightarrow prog_i) \\
\quad \Box\ b1 \vee b2 \rightarrow prog \\
\quad \textbf{fi}
\end{array}
$$

線形コード (変形9)

$$
\begin{array}{ll}
\textbf{if} & d = 0 \vee a[i] = x \ \rightarrow\ x, d \ :=\ a[i], d+1 \\
\Box & d \neq 0 \wedge a[i] \neq x \ \rightarrow\ d \ :=\ d \ominus 1 \\
\textbf{fi} &
\end{array}
$$

変形

以上でかなり簡単になった．これを **if** $\cdots$ **then** $\cdots$ **else** $\cdots$ **end** 構文によるプログラムとして書いたものがつぎである．

最終コード

$$
\begin{array}{l}
[\![\ \ \textbf{var}\ i, d\ :\ \mathbb{N} \bullet \\
\\
\quad i, d\ :=\ 0, 0\ ; \\
\quad \textbf{do}\quad i \neq N\ \rightarrow \\
\qquad \textbf{if}\quad d = 0 \vee a[i] = x \\
\qquad \textbf{then}\ x, d\ :=\ a[i], d + 1 \\
\qquad \textbf{else}\ d\ :=\ d \ominus 1 \\
\qquad \textbf{fi}\ ; \\
\qquad i\ :=\ i + 1 \\
\quad \textbf{od} \\
]\!]
\end{array}
$$

参考文献

⋆J.-R. Abrial, The B-Book Assigning Programs to Meanings. CU 1996.

⋆⋆R.-J. Back and J. von Wright, Refinement Calculus A Systemat Introduction. Springer, 1998.

⋆E. W. Dijkstra, A Discipline of Programming. Prentice Hall, 197 土居, 原田訳, プログラミング原論. サイエンス社, 1983.

⋆D. Gries, The Science of Programming. Springer, 1981. 筧訳, プログラミングの科学. 培風館, 1991.

⋆林晋, プログラム検証論. 共立出版, 1995.

⋆⋆C. Morgan, Programming from Specifications(2nd ed.). Prentic Hall, 1994.

⋆⋆C. Morgan and T. Vickers(Eds.), On the Refinement Calculus Springer, 1992.