創発的計算のための言語 SOOC: その特徴と実装

— 魔方陣を例題として —

新情報処理開発機構　つくば研究センタ

金田　泰

目次

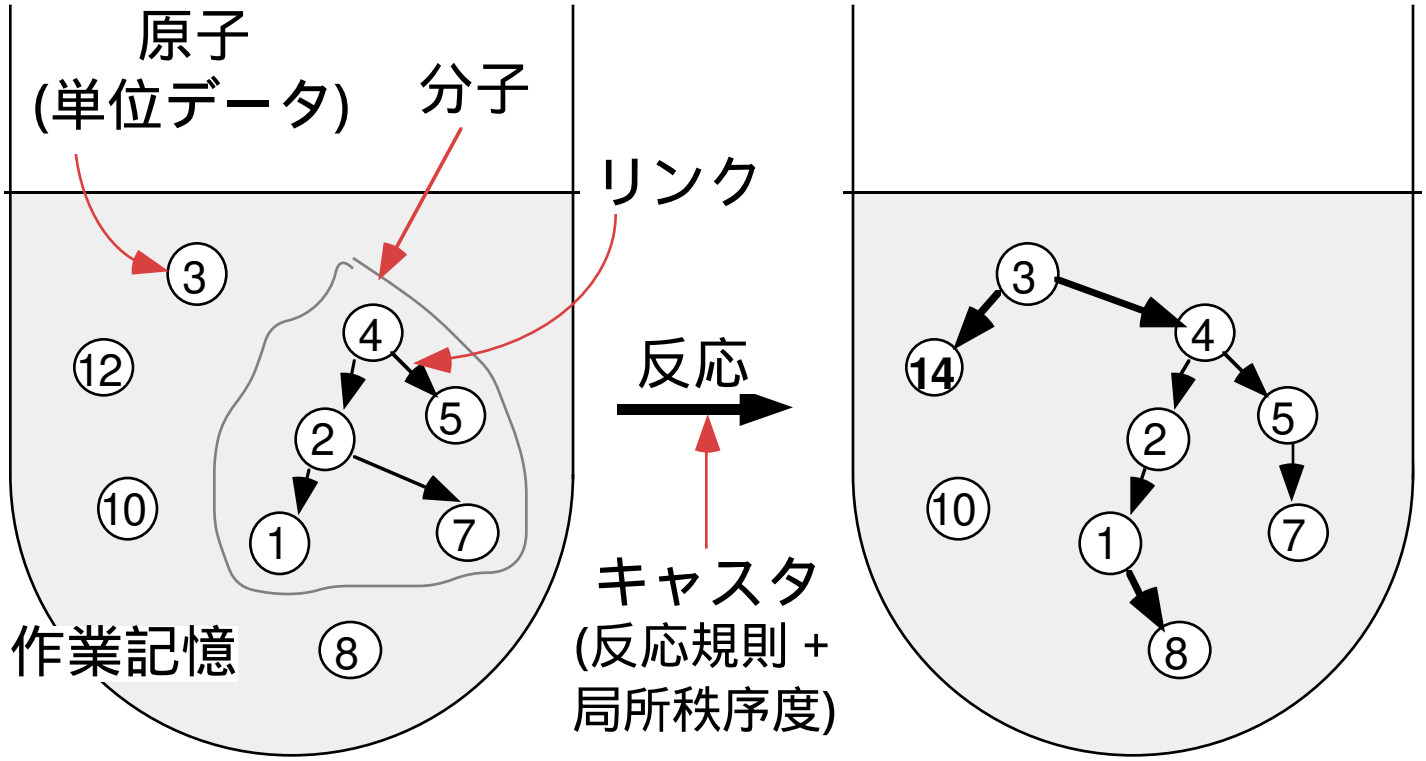
- はじめに
- 創発的計算のためのモデル CCM
- CCM にもとづく言語 **SOOC-94** (魔方陣を例題として)
- **SOOC-94** の特徴
- **SOOC-94** の実装
- 結論

はじめに

- 実世界 (の問題) は
  - 複雑である．
    - 非線形であり，独立なモジュールへの分割ができない
      (“全体” は部分の和をこえたもの) .
  - 環境 (人間，自然) に対してひらかれている．
    - 不意の入力に耐える必要がある (追従性, 短期的要求) .
    - 環境や問題の変化に適応する必要がある (適応性, 長期的要求) .
- 実世界の問題をとくための方法論を確立することが研究目標．
  - 創発的計算が鍵になる (?) — そのためのモデル CCM を提案している．
- 創発的計算 (Emergent Computation)
  - 局所的・部分的な情報だけで計算し，大域的・全体的な結果 (秩序状態) をうみだす自己組織的な計算．
  - あたえられた情報からは予測できないような結果をえる計算．
  - 決定論的ではない — バクチ的 (?)

創発的計算のためのモデル CCM — 1

- 創発的計算のためのモデルとして CCM を提案している．
  - CCM はプロダクション・システムにもとづく．
  - CCM = Chemical Casting Model (化学的キャスティング・モデル)．
  - “化学的”: 化学反応系とのアナロジーをつかっている．
  - Casting: 従来のことばでは「プログラミング」or「計算」．

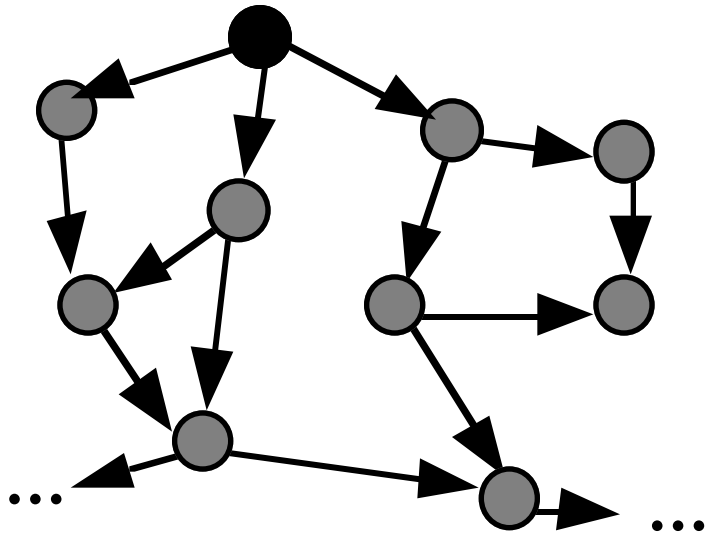


創発的計算のためのモデル CCM —2

- CCM が従来のプロダクション・システムとことなる点
  - 局所秩序度 (局所評価関数) にもとづいて動作
    - ニューラルネットのエネルギー関数や GA の適応度関数は大域評価関数．
    - 局所秩序度は原子間結合エネルギーに負号をつけたようなもの．
    - 秩序度が増加するときだけプロダクション規則が動作．
    - やまのぼりではない．
      - 秩序度の増加方向に動作するが，局所秩序度は大域評価関数ではないから．
  - 確率的 (ランダム) に動作 — 並列 / 協調 動作の可能性がある．
- 局所的な情報だけをつかって計算し，大域的な結果をえることをめざす — 創発的計算をめざす．

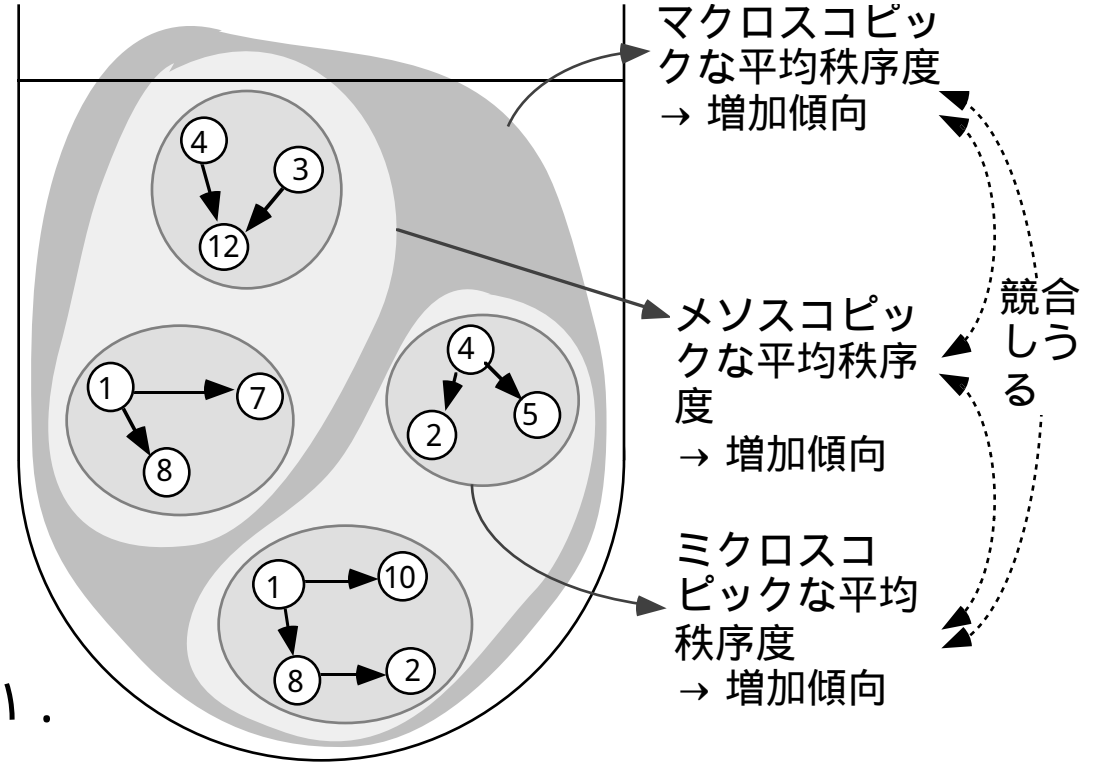
CCM にもとづく計算と問題解決*

- CCM による計算はバイアスがかかったランダムな近傍探索とみなせる [SWoPP 93]
  - 反応規則 (プロダクション規則)
    近傍の状態 (遷移可能な状態) をきめる — 探索のおおわくをきめる．
  - 局所秩序度 (局所評価関数)
    局所的にみて “よりよい” 状態を定義 — バイアスをきめる．
- CCM にもとづく問題解決
  - “よりよい” 状態としてよりおおくの制約がみたされた状態をとる
    — 制約充足問題がとける．
  - “よりよい” 状態としてより最適化された状態をとる
    — 最適化問題の近似解がもとめられる．



平均秩序度

- 平均秩序度 = 局所秩序度の平均値
  - システムは平均秩序度が増加する方向に揺動的 (stochastic) に動作．
    - 一時的には減少することもある．
  - ただし，平均秩序度はシステムの動作にかかわらない．
    - 平均秩序度はシステムを外部から観測するときにつかう．
- 平均秩序度はミクロなレベルでもマクロなレベルでも定義できる．
- 平均秩序度は開放系でも定義できる
  - 以前つかっていた大域秩序度 (局所秩序度の総和) は閉鎖系でしかつかえない．



例題：魔方陣の問題と解法

■ 問題

- ◆ 魔方陣とは：$N \times N$ のますに 1 から N^2 までの整数値をいれて各行各列および対角線上の数の和がすべて一定値 S になるようにした方陣．
- ◆ 3 次 ($N = 3$) の魔方陣の例：($S = 15$)

2	9	4
7	5	3
6	1	8

- ◆ 魔方陣をもとめる制約充足問題をとく．

■ 解法の概略

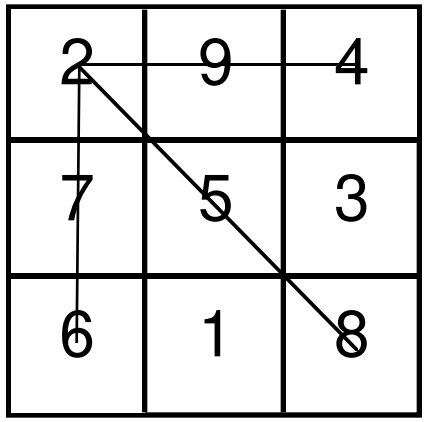
- ◆ まず，用意した $N \times N$ の方陣に 1 ~ N^2 の整数を適当に配置する．
  - 整数をならべる順序：昇順, 降順, or ランダム．
- ◆ 2 個の整数を交換する操作をくりかえして解をもとめる．

CCM にもとづく計算言語 SOOC-94

- **SOOC-94 は Lisp 風の言語で，Lisp 上に実装されている．**
- 目次
  - データ型の定義
  - 局所秩序度の定義
  - 反応規則の定義
  - 初期設定と起動

データ型の定義 — 1

CCM にもとづく計算言語 SOOC-94



■ 魔方陣の例

◆ 各ます (column) を原子としてあつかう．

```
(defelement column
  value □□                    ; 整数値
  (* summation)□               ; 非排他要素 (Lisp にない構文)
  □     □ ; このますが属する行，列，対角線の和に
  □     □ ; 関する 0 個以上のデータ．
  x  y) □ □                    ; 座標 (印刷用)
```

◆ マクロ defelement は Lisp の defstruct にちかいが拡張されている．

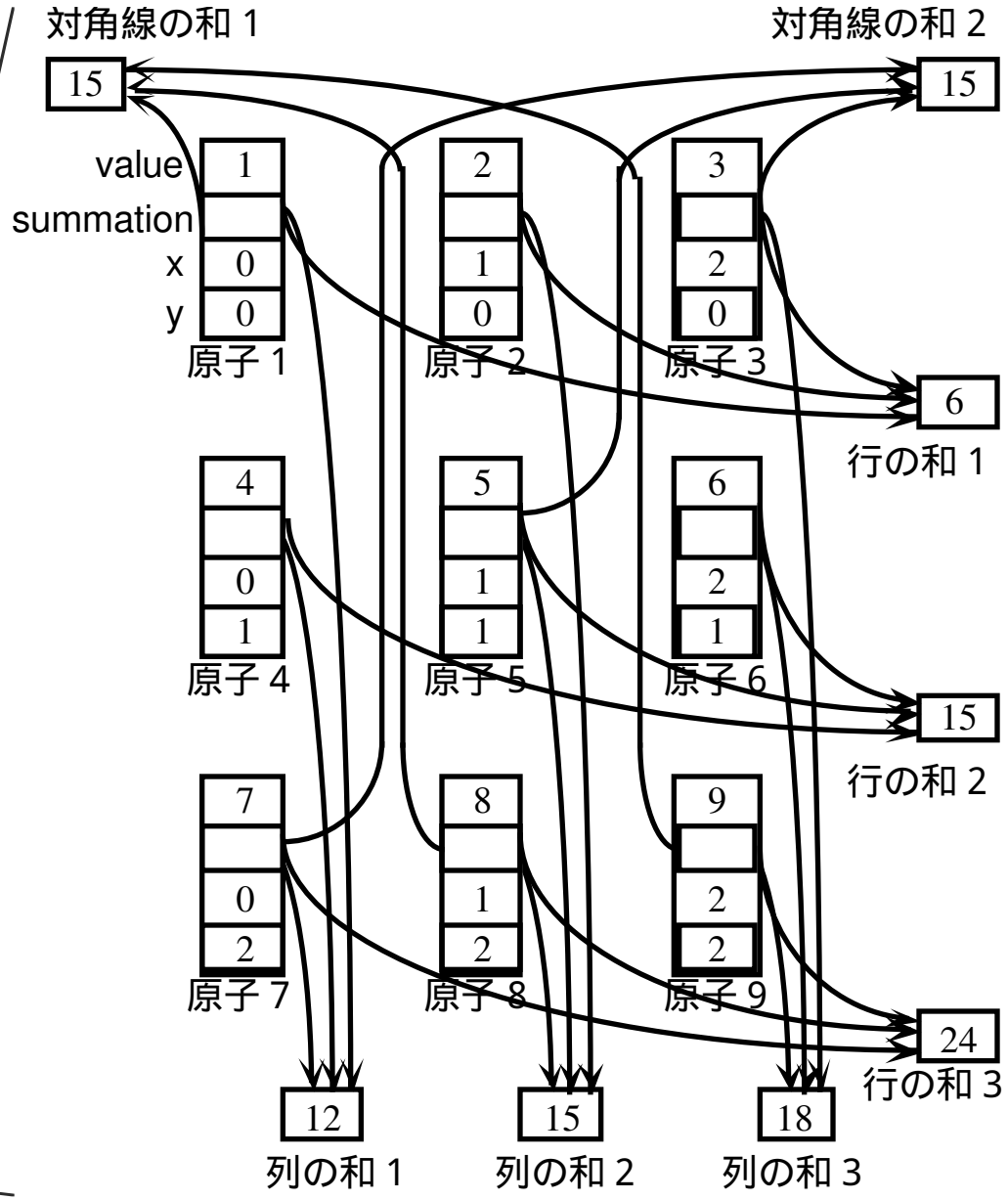
◆ 非排他要素： summation という名称の要素は 0 個以上の任意個存在することがゆるされる (同名の他の要素を排斥しない)．

データ型の定義 — 2

CCM にもとづく
計算言語 SOOC-94

- ひとつの魔方陣全体をあらわすデータ構造
  - column 型の原子 9 個と行，列，対角線の和をあらわすデータ (原子 ?) — リンクで結合．
  - 初期設定でつくる．

2	9	4
7	5	3
6	1	8



局所秩序度の定義 — 1

CCM にもとづく計算言語 SOOC-94

- 制約充足問題の局所秩序度の基本
  - 各制約がみたされていれば 1，みたされていなければ 0 となるように局所秩序度を定義 [SIGSYM 93.3]．
- 魔方陣 — 基本とちがう局所秩序度を定義
  - 各ますについて 2 値ではない局所秩序度 $o(C)$ を定義する．
  - 制約 c_i ($i = 1, 2, \ldots, m$): ますが属するある行，列，または対角線に関する和 S_i が S でなければならない．
  - 制約 c_i に対応する秩序度: $o_i = -(S - S_i)^2$
    - すべての和が S であるときは 0 ，それからはずれるにつれて減少．
  - $o(C) = \sum_{i=1}^{m} o_i$
  - システムは $o(C)$ の平均値 (平均秩序度) が増加する方向に動作
    — 解にむかって動作．

局所秩序度の定義 — 2

CCM にもとづく計算言語 SOOC-94

■ 魔方陣の局所秩序度の表現

◆ (deforder ((c column))　　; 自己秩序度 (⇔ 相互秩序度)
  (let ((sum 0))
    (do-* (column c :summation t-sum)　　—— do−*
      (let ((diff (– *expected-sum* (summation-value t-sum))))
        (decf sum (* diff diff))))
  sum))

- deforder マクロは CLOS の defmethod マクロにちかい．
- マクロ do-* の意味
  - 同名の非排他要素全部に対して演算する．
  - do−*：column 型の原子 c の summation という名の要素の値を t-sum という変数に束縛して，くりかえし評価．

反応規則の定義

CCM にもとづく計算言語 SOOC-94

- 反応規則の機能
  - 2 つのますがふくむ整数値を交換すること
  - 交換にともなって行，列，対角線の和の値を更新
- 定義
  - (defrule magic-square ; 反応規則名
    (var V1 V2 C1 C2) ; 変数宣言
    (exist column C1 :value V1) ; 左辺のマッチング・パタン 1
    (exist column C2 :value V2) ; 左辺のマッチング・パタン 2
    —> ; (排他マッチング)
    (exist column C1 :value V2) ; 右辺のマッチング・パタン 1
    (exist column C2 :value V1) ; 右辺のマッチング・パタン 2
    (action (update-summations C1 (– V2 V1))
    (update-summations C1 (– V1 V2))) ; 動作定義 1 (Backtrack)
    (action (update-summations C2 (– V1 V2))
    (update-summations C2 (– V2 V1)))) ; 動作定義 2

初期設定と起動

CCM にもとづく計算言語 SOOC-94

- 現在はユーザが Lisp によって初期設定をしてから起動する．
  - 原子がない状態から反応規則をつかって初期設定するのは困難だから．
  - ただし，初期設定の際に make，modify，exist などの SOOC-94 のコマンドをつかうことができる．
  - 将来は SOOC によって記述できる範囲を拡大したい．
- 魔方陣のシステムを起動するための Lisp プログラム
  - (defun magic-square (n)　　; n は次数
    (setq *N* n)
    (init)　　; SOOC-94 システムの初期設定
    (make-square n)　　; データ構造の初期設定
    　　; (make-square はユーザ定義の関数．なかで make などを使用)
    (run　; SOOC-94 システムを起動する．
    :global-strategy random　; 大域的なスケジュリング戦略の指定
    :rules (magic-square)　; 使用する規則の集合の指定
    :max-reaction-times (* 100 n n)))
    　　　　; 計算うちきり条件 (テスト回数) の指定

SOOC-94 の特徴

目次

- 秩序度の自動計算とバックトラック
- 反応規則の両辺の構文の統一
- スケジューリング戦略とその指定
- 原子の非排他要素

秩序度の自動計算とバックトラック

SOOC-94 の特徴 — 1

- 秩序度の自動的計算
  - 反応をおこすかどうかをきめるために局所秩序度の局所的な和が計算される．
  - 反応後の秩序度の計算のためには，一般には実際に反応をおこしてみる必要がある．
- バックトラック
  - 反応をおこさないときめたとき，反応の効果を無効にする．
    - 反応がおこらなかったことにする．
  - 局所的なバックトラックが必要だという点は並列論理型言語に似ている．
    - 実装上の問題：外部からみえてはならない計算結果が輸出されるという問題がおこりうる．

反応規則の両辺の構文の統一

SOOC-94 の特徴 — 2

- **SOOC-94** においては極力，両辺の構文を統一している．
  - 右辺のパタンの例： (exist column C1 :value V2)　— 左辺と同構文
- 通常のプロダクション・システムでは両辺の構文はことなる．
  - 右辺のコマンドの例： (modify C1 :value V2)
- 両辺の構文を統一しようとする理由
  - 本来 CCM においては反応規則が可逆．
    - 反応規則の右辺を条件部，左辺を動作部としてもつかえる．
  - 通常は実行が評価関数のようなもので制御されていないから．
    - 規則を逆むきにつかうことをゆるすと停止しなくなる．
  - 柔軟な計算のためには可逆であるのがよい．
    - 環境の変化によって平衡点がかわるような計算が実現しやすい．
- 同一表現からことなるコードを生成する必要がある．

スケジューリング戦略とその指定

SOOC-94 の特徴 — 3

- スケジューリング戦略の選択によってまったくことなる制御構造をもつコードが生成される．
  - スケジューリングとは反応の順序をきめること．
  - 複数のスケジューリング戦略のなかから適当なものを選択できる．
- 2 種類のスケジューリング戦略
  - ランダム戦略
    - 反応に使用する規則と対象データとはランダムに選択する (標準)
    - 指定法： (defrule (rule-name :strategy random) …)　— CLOS 的
  - 系統的な戦略
    - すべての原子の順列を順に生成し，それにもとづいて原子を選択．
    - 指定例： (defrule (rule-name :strategy depth-first) …)
  - 戦略の比較
    - 系統的な戦略はリミット・サイクル (Looping) をおこしうる．
    - 系統的な戦略が必要またはそのほうがよりよいばあいもある．

原子の非排他要素

SOOC-94 の特徴 — 4

- 原子が非排他要素をもちうることが **SOOC-94** の特徴のひとつ
  - 非排他要素 : 複数の同名の要素の存在がゆるされるような要素 .
  - (defelement column
    value □□ ; 整数値
    (* summation)□ ; 非排他要素
    x y) □ □ ; 座標 (印刷用)
- 非排他要素をとりいれた理由
  - 任意個のリンクをもつ原子をあつかう反応規則が非常に容易，美的に記述できるから .
  - 配列のようにいちいち添字を指定する必要がない .
  - 非排他要素のなかの 1 個だけをランダムに選択してつかう反応規則においては利点がおおきい (たとえば彩色問題のとき) .

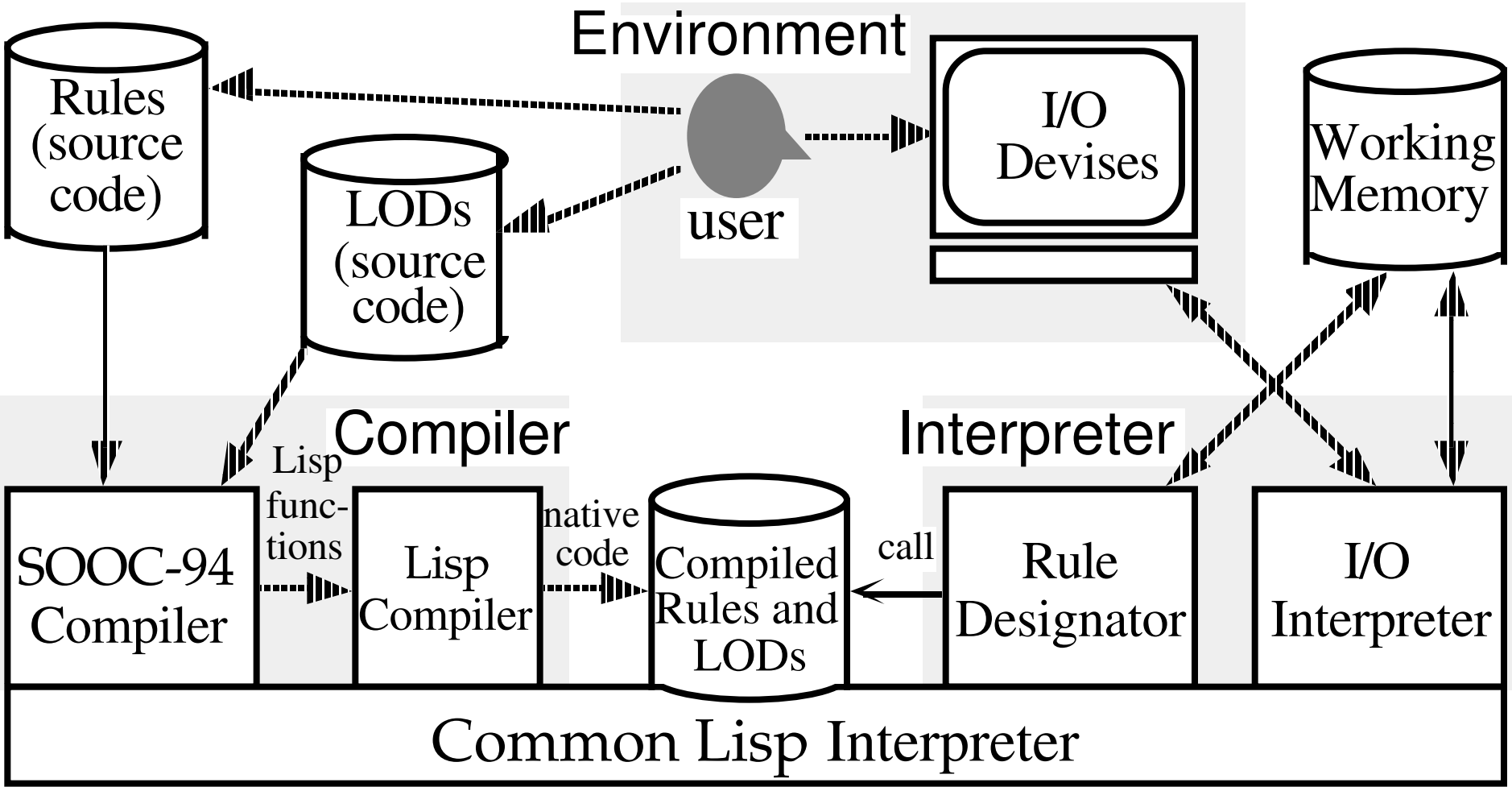
逐次計算機のための SOOC-94 の実装

目次

- 処理系の基本構造
- コンパイル例
  - データ型定義
  - 局所秩序度定義
  - 反応規則定義
- 作業記憶の視覚化機能

処理系の基本構造

SOOC-94 の実装 — 1



データ型定義のコンパイル例

SOOC-94 の実装 — 2-1

■ 魔方陣の原始プログラム

◆
```
(defelement column
  value
  (* summation)                    ; 非排他要素
  x  y)
```

■ 魔方陣の目的プログラム

◆
```
(progn
 (fmakunbound 'column-ord)□ 局所秩序度のふるい定義を削除．
 (defstruct column
  value □; 整数値
  (summation (sooc94::make-empty-array))
□   □          ; 空の拡張可能配列を要素 summation の初期値として
□   □    ; 設定している (非排他要素は配列として実装)．
  x  y) □   ; 印刷用の座標．
 (set-pprint-dispatch 'column (formatter "~/sooc::pprint-atom/")))
```

局所秩序度定義のコンパイル例

SOOC-94 の実装 — 2-2

- 魔方陣の原始プログラム
  - ```
    (deforder ((c column))
      (let ((sum 0))
        (do-* (column c :summation summation)
          (let ((diff (– *expected-sum* (summation-value summation))))
            (decf sum  (* diff diff))))
      sum))
    ```
- 魔方陣の目的プログラム
  - ```
    (defun column-ord (c)
     (let ((sum 0))
      (let ((t24 (column-summation c)))□   ; この行以下は do-* の展開形 .
    □   □   ; t24 は拡張可能な配列 (非排他要素は配列として実装) .
       (dotimes (t25 (length t24) t) □      ; 非排他要素を順にアクセス
         (let ((summation (aref t24 t25)))
          (let ((diff (– *expected-summation*
               (summation-value summation))))
           (decf sum (* diff diff))))))
      sum))
    ```

反応規則定義のコンパイル例 (魔方陣目的プログラム)

SOOC-94 の実装 — 2-3

■
```
(defun run-magic-square ()
  (let ((sooc94::*rule-name* 'magic-square))
    (incf sooc94::*ntests*)
    (block sooc94::body
      (let ((c2 'sooc94::<<unbound>>) (c1 'sooc94::<<unbound>>)
            (v2 'sooc94::<<unbound>>) (v1 'sooc94::<<unbound>>))      ; 変数値の初期設定
        (unless (and (exist column c1 :value v1)        ; 左辺のパタン 1 に由来するマッチング・パタン．
                     □                                  ;  c1, v1 に値が代入される．
                     (exist column c2 :value v2)        ; 左辺のパタン 2 に由来するマッチング・パタン．
                     □                                  ;  c2, v2 に値が代入される．
                     (not (eq c2 c1)))                  ; c1, c2 にことなる原子がマッチすることを保証．
          (return-from sooc94::body nil))
        (without-interrupts                             ; 動作のただしさを保証するためにわりこみを禁止．
          (let ((sooc94::_order (+ (column-ord c2) (column-ord c1)))
                (t26 (column-value c1))  (t27 (column-value c2)))
            (modify-basic column c1 :value v2) □        ; 右辺のパタン 1 に由来する動作．
            (modify-basic column c2 :value v1) □        ; 右辺のパタン 2 に由来する動作．
            (update-summations c1 (– v2 v1))            ; 右辺の動作定義 1 に由来する動作．
            (update-summations c2 (– v1 v2))            ; 右辺の動作定義 2 に由来する動作．
            (unless (<= sooc94::_order (+ (column-ord c2) (column-ord c1)))
                                                        ; 秩序度に関する条件が不成立ならば (バックトラックする)
              (update-summations c2 (– v2 v1))          ; 右辺の動作定義 2 に由来する逆動作．
              (update-summations c1 (– v1 v2))          ; 右辺の動作定義 1 に由来する逆動作．
              (modify-basic column c2 :value t27)       ; 右辺のパタン 2 に由来する逆動作．
              (modify-basic column c1 :value t26)       ; 右辺のパタン 1 に由来する逆動作．
              (return-from sooc94::body nil)))
          (setq *reaction-times* 0)                     ; 左辺テストのカウンタリセット
          (incf sooc94::*nactions*))))))
```

作業記憶の視覚化機能

SOOC-94 の実装 — 3

- 反応ごとに，かきかえられた原子を自動的に表示する．
  - 具体的には make，modify などのコマンドが実行されるごとに表示．
  - 表示のためにデータを defelement で確保．
  - 実行開始時にウィンドウをひらく．
- ユーザが指定するもの
  - ウィンドウの初期設定パラメタ
  - 各データ型ごとの表示ルーティン (と消去ルーティン)
  - 魔方陣のばあいには，あわせて 30 行程度．
- 視覚化機能の使用目的
  - 計算結果の表示．
  - 計算過程の観測．

結言

- CCM に関する実験をおこなうための計算言語 **SOOC-94** について，その特徴と実装を中心として説明した．
- 報告した特徴はいずれも **SOOC-94** に特有のもの
  - すぐには他の言語に応用できない．
  - 今後さらに発展させるに値する．
- 将来はより大規模な実験をおこないたい．
  - 現在の SOOC は大規模応用システムの記述には適さないので，改良が必要．
- 当面の研究課題
  - 並列計算機への実装
  - 計算や探索の局所性を制御する手法の実装
    - 反応規則の自動合成など

*

- 制約充足問題において反応規則によって正当性をたしかめたいとき：ランダム戦略をつかうとチェックされない制約がのこりうる．
- 系統的な戦略のほうが高速μリミット・サイクルの確率が十分ちいさくおさえられ，ひくいコストでその検出ができるばあいはよりよい．