プロダクションシステム（Ｉ）

The Atomic Components of Thought, John R. Anderson (著), Christian Lebiere (著) (1998/06) Lawrence Erlbaum Assoc Inc

1. 概要、知識表現　pp.1~55
2. 実行（パフォーマンス）　pp.57~100
3. 学習　pp.101~142

計算モデルとしてのプロダクションシステム

ガイダンス資料より

認知科学の特徴

- 情報処理的アプローチ
  - 心の働きを情報の流れとしてみる
  - コンピュータにアナロジー
    ⇒　メモリ、プロセッサ、プログラム
- 計算論的アプローチ
  - モデル
  - シミュレーション
  - 予測、説明
  - 近似、適用限界
- 学際性
  - 心理学、工学、人工知能、言語学、人類学、計算機科学、哲学、神経科学

計算モデルとしてのプロダクションシステム

ガイダンス資料より

シミュレーションとは

ハノイの塔を解くのに必要な手続き知識の一部

```
;;; the first component of the perceptual strategy (Simon 1975's production
;;; P7') is to identify the largest out-of-place (LOOP) disk.  this can be done
;;; perceptually, but here we store it in the goal chunk (largest).  when
;;; the disk we just moved (moved) is also the largest, update largest to be
;;; the next smaller disk, then focus on that (focus-on-largest production).

(p update-largest-after-move
   =focus>
      isa attend
      largest =size     ; the LOOP disk is
      moved =size       ;   now the disk we just moved
    - largest zero

   =fact>
      isa countfact
      first =new
      then =size
==>
   =focus>
      largest =new      ; so update the LOOP disk
      skipped nil       ; disable retrieve's; want a new plan now, not an old one.
      count (!eval! *subgoal-iterations*) ; see check-strength

   !output! ("The largest disk out of place is now ~S" =new)
)
```

シミュレーションの
トレース

```
Trials done in 15 steps ..... 977 (out of 2000)
Average moves per trial ..... 17.7
Retrieval threshold ..... 4.0
Activation noise ..... 0.3
Subgoal iterations ..... 2

Move Number     Time (sec.)
     1            9.00
     2            2.20
     3            2.55
     4            2.21
```

PSは、人間の認知を統一的に理解するための試みとして誕生

- 人間の認知を扱うアカデミックな研究のDivide-and-Conquer 戦略をとることによる蛸壺化
  - 伝統的な心理学ではデータが尊重され、データから、精密な反応構造（何が）と時間構造（いつ）を導き出すことが要求される　→　現象を細分化しないと要求にこたえられない
  - 現象を細分化して個々のものを理解し、その後、それらをまとめて全体が理解できると信じられていた
- Newell の見解, 1972 Carnegie Symposium
  - Each seemed to manufacture an endless stream of research without the overall picture of human cognition becoming any clearer.
  - Unified Theories の開発による状況の打開
    - 認知の全ての側面を扱うことのできる理論
    - 認知心理学における人間の認知を科学的に理解する為の新しいアプローチ法　→　プロダクションシステム
- 補足：モデルヒューマンプロセッサも人間の認知行動を統一的に記述するものであるが、それは“応用”認知心理学という特徴を持っている。人間の認知を科学的に理解することではなく、認知心理学の知見を工学的に応用するための枠組みを与えている

Newell’s first production system theory of human cognition (Newell, 1973)

A production system is a scheme for specifying an information processing system. It consists of a set of productions, each production consisting of a condition and an action. It has also a collection of data structures: expressions that encode the information upon which the production system works – on which the actions operate and on which the conditions can be determined to be true or false.

A production system, starting with an initially given set of data structures, operates as follows. That production whose condition is true of the current data (assume there is only one) is executed, that is, the action is taken. The result is to modify the current data structures. This leads in the next instant to another (possibly the same) production being executed, leading to still further modification. So it goes, action after action being taken to carry out an entire program of processing, each evoked by its condition becoming true of the momentarily current collection of data structures. The entire process halts either when no condition is true (hence nothing is evoked) or when an action containing a stop operation occurs.

Newell’s first production system theory of human cognition (Newell, 1973), Cont.

Much remains to be specified in the above scheme to yield a definite information processing system. What happens (a likely occurrence) if more than one production is satisfied at once? What is the actual scheme for encoding information? What sort of collection of data structures constitutes the current state of knowledge on which the system works? What sort of tests are expressible in the conditions of productions? What sort of primitive operations are performable on the data and what collections of these are expressible in the actions of productions? What sorts of additional memories are available and how are they accessed and written into? How is the production system itself modified from within, or is it possible? How much time (or effort) is taken by the various components of the system and how do they combine to yield a total time for an entire processing?

Newell’s first production system theory of human cognitionの概要

- 情報処理システムを詳述する為の図式
- 構成要素
  - プロダクション
    - 「条件部」と「実行部」から成る
  - データ構造
    - 符号化された情報
      - ✓ プロダクションの実行部が作用する
      - ✓ プロダクションの条件部の真偽が決定される
- 動作
  1. 初期状態のデータ構造からはじめる
  2. データ構造に条件部が照合するプロダクションの実行部が実行される
  3. データ構造が更新される
  4. データ構造に条件部が照合するプロダクションの実行部が実行される
     ・・・
     （繰り返し）
     ・・・
  5. 停止：「どのプロダクションも条件が満たされない」または「停止する」という実行部をもつプロダクションが実行される

Newell’s first production system theory of human cognitionの概要

- 情報処理システムとして動かす為にはまだ詰めなければならない点がある

1. 複数のプロダクションが条件を満たした場合、どうすればいいのか
2. 情報の符号化は、実際にはどのように行えばいいのか
3. 知識の状態（プロダクションシステムが作用する）として、どのようなデータ構造を準備しておけばいいのだろうか
4. プロダクションの条件部について、照合テストはどのように表現できるだろうか
5. データ構造の上でどのような基本オペレータが作用するのだろうか
6. プロダクションの実行部について、オペレータをどのようにまとめて、実行部として表現できるだろうか
7. 記憶として他に利用できるものがあるだろうか。また、アクセス、書き込みはどのように行えるだろうか
8. プロダクションシステムの自発的更新はどのようになされるだろうか
9. 照合時間、実行時間など、個々の処理時間はどれくらいだろうか
10. 個々の時間から、どのようにして全体処理時間が導き出せるだろうか

現在までに提案されているプロダクションシステム理論

- プロダクションシステムは、幅広いタスクの遂行を司る複雑な認知過程を、十分な詳細さかつ精度でモデル化できる唯一の形式

- ACT-R (Anderson, 1993)
  - 長期記憶の利用に着眼し、実行過程ばかりでなく、学習も扱う。現在、最も成功している理論
- 3CAPS (Just & Carpenter, 1992)
  - 作業記憶の利用メカニズムに着眼した理論
- EPIC (Meyer & Kieras, 1997)
  - MHPのプロダクションシステム版。認知過程ばかりでなく、知覚過程と運動過程も理論に含まれている
- Soar (Newell, 1991)
  - 複数の問題空間を用意し、各問題空間内と問題空間間の間の遷移で認知行動をモデル化する理論
- Construction-Integration Theory (Kintsch, 1992)
  - 理解するプロセスが認知の中心にあると考え、主として文章理解を扱う

ACT-R を構成する理論

1. 知識の性質に関する理論
   - ▶ 知識の種類、知識表現
     - ✓ 情報の符号化は、実際にはどのように行えばいいのか
     - ✓ 知識の状態（プロダクションシステムが作用する）として、どのようなデータ構造を準備しておけばいいのだろうか
     - ✓ プロダクションの条件部について、照合テストはどのように表現できるだろうか
     - ✓ データ構造の上でどのような基本オペレータが作用するのだろうか
     - ✓ プロダクションの実行部について、オペレータをどのようにまとめて、実行部として表現できるだろうか
     - ✓ 記憶として他に利用できるものがあるだろうか。また、アクセス、書き込みはどのように行えるだろうか
2. 知識の利用のされ方に関する理論
   - ▶ パフォーマンス（行為選択、実行時間）
     - ✓ 複数のプロダクションが条件を満たした場合、どうすればいいのか
     - ✓ 照合時間、実行時間など、個々の処理時間はどれくらいだろうか
     - ✓ 個々の時間から、どのようにして全体処理時間が導き出せるだろうか
3. 知識の獲得のされ方に関する理論
   - ▶ 学習
     - ✓ プロダクションシステムの自発的更新はどのようになされるだろうか

知識の性質　・・・　2種類の知識

1. 宣言的知識
   Declarative knowledge
   - 思い出すことができて他の人に伝達できるとわかっていることがらに関する知識
     - 日本サッカーがワールドカップに初めて参加したのはフランス大会だ
     - 3と4の和は7である

2. 手続き的知識
   Procedural Knowledge
   - 行動に現れるが自覚していない知識

ACT-R における宣言的知識

- 宣言的知識は、チャンクとして表現される
- 例：“3と4の和は7である”

チャンクの名前であり、任意

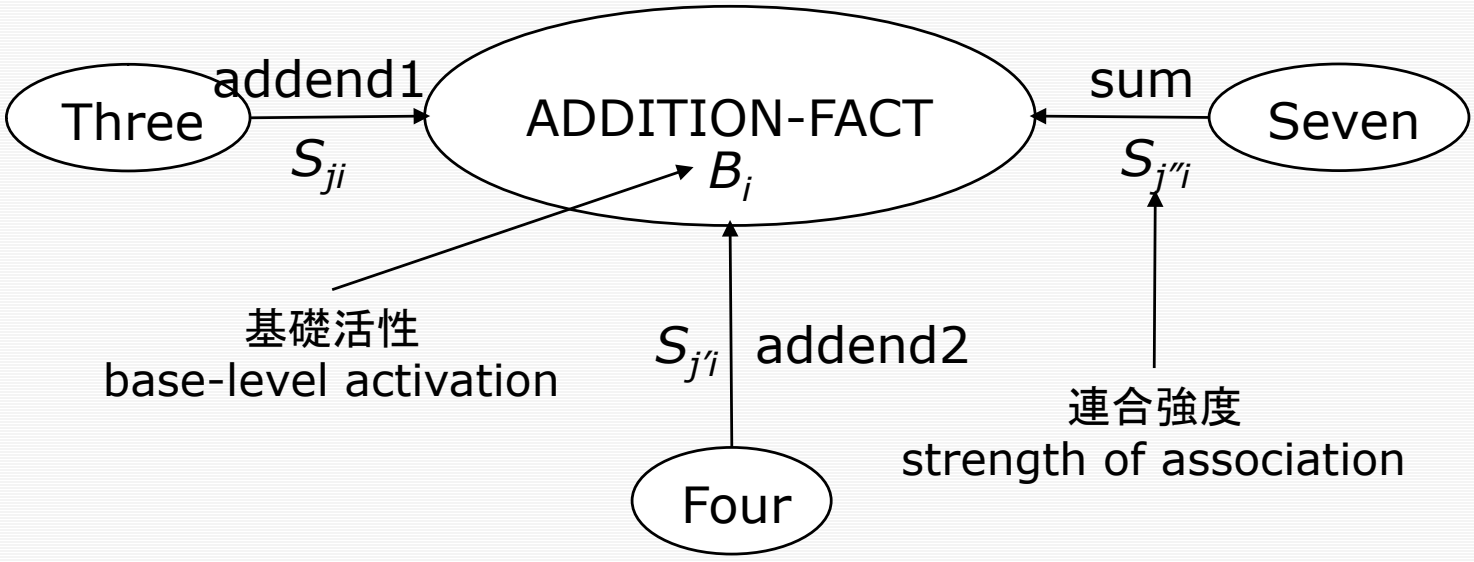
テキスト表現

isa: チャンクのタイプを指定する特別なスロット

Fact3+4	
isa	ADDITION-FACT
addend1	Three
addend2	Four
sum	Seven

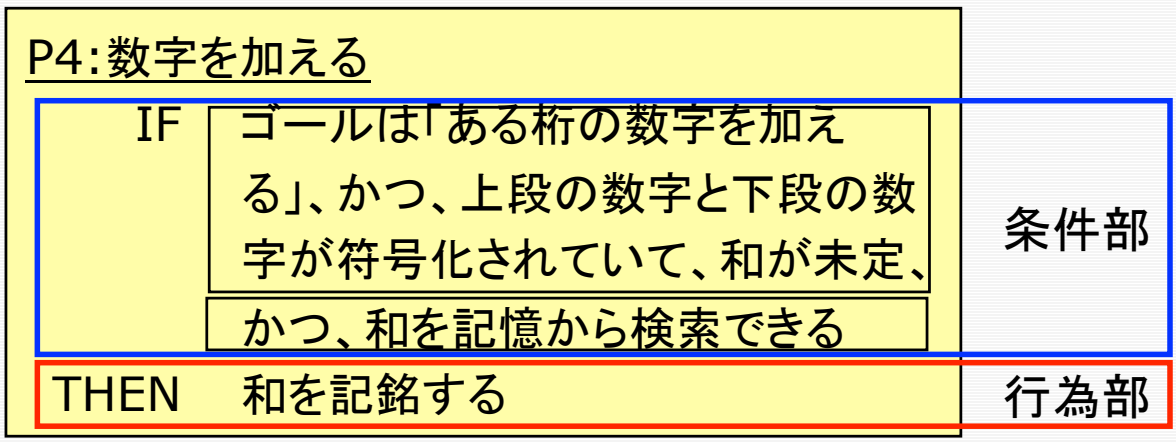
addition-factに固有のスロットと値

ネットワーク表現



ACT-R における手続き的知識

1. 手続き的知識は、宣言的知識を検索して利用する方法を記述するプロダクションルールとして表現される
2. プロダクションルールは「条件-行為」の対で表現される　・・・　Condition–Action pair
3. 「条件部」はそのプロダクションルールが適用される為の条件を記述する
4. 「行為部」はプロダクションが適用されたときに行うべき事柄を記述する
5. 現在のゴールの状態が照合し、かつ、条件部に記述されている宣言的知識が検索可能なプロダクションが適用される
6. プロダクションを適用することにより、ゴール状態を変更することが可能である



プロダクションを順次発火させて、高次の認知（例えば、足し算）を達成

- Cycle 0: P1.開始する
  - Cycle 1: P2.上段の数字を読む
  - Cycle 2: P3.下段の数字を読む
  - Cycle 3: P4.数字を加える
  - Cycle 4: P5.答えを出す
  - Cycle 5: P7.答えを書く

4

- Cycle 6: P10.次の桁に進む
  - Cycle 7: P2.上段の数字を読む
  - Cycle 8: P3.下段の数字を読む
  - Cycle 9: P4.数字を加える
  - Cycle 10: P6.繰り上がりを処理する
  - Cycle 11: P5.答えを出す
  - Cycle 12: P7.答えを書く

6

- Cycle 13: P10.次の桁に進む
  - Cycle 14: P2.上段の数字を読む
  - Cycle 15: P3.下段の数字を読む
  - Cycle 16: P4.数字を加える
  - Cycle 17: P5.答えを出す
  - Cycle 18: P7.答えを書く

3

- Cycle 19: P10.次の桁に進む
  - Cycle 20: P3.下段の数字を読む
  - Cycle 21: P8.最後の桁を処理する（繰り上げなし）
- Cycle 22: P11.終了する

```
  ２ ３ ９
＋ １ ２ ５
―――――――
```

2桁2数字足し算問題のプロダクションルール（テキスト版）

P1:開始する
IF ゴールは「足し算を行う」、かつ、処理する桁が決まっていない
THEN サブゴール「一の桁の数字を加える」を設定する、そして、「次に処理する桁は十の桁」を記銘する

P2:上段の数字を読む
IF ゴールは「ある桁の数字を加える」、かつ、上段数字が符号化されていない
THEN その桁の上段の数字を符号化する

P3:下段の数字を読む
IF ゴールは「ある桁の数字を加える」、かつ、下段の数字が符号化されていない
THEN その桁の下段の数字を符号化する

P4:数字を加える
IF ゴールは「ある桁の数字を加える」、かつ、上段の数字と下段の数字が符号化されていて、和が未定、かつ、和を記憶から検索できる
THEN 和を記銘する

P5:答えを出す
IF ゴールは「ある桁の数字を加える」、かつ、和が計算済み、かつ、和に一の桁の数字と十の桁の数字がある
THEN 「十の桁の数字を繰り上げ数字」として記銘、そして、一の桁の数字を答えに設定する

P6:繰り上がりを処理する
IF ゴールは「ある桁の数字を加える」、かつ、答えがある、かつ、繰上げ数字がある
THEN 答えを一つ増加させた数字に変更する、そして、繰上げ数字があるというマークを削除する

P7:答えを書く
IF ゴールは「ある桁の数字を加える」、かつ、繰上げ数字がない、かつ、繰り上げるべき数字が決定している
THEN 答えを解答行に書く、そして、ゴールをポップする

P8:最後の桁を処理する（繰り上げなし）
IF ゴールは「ある桁の数字を加える」、かつ、下段の数字が読み込み済み、かつ、それが＋である、かつ、繰上げ数字がない
THEN 「問題終了」を記銘、そして、ゴールをポップする

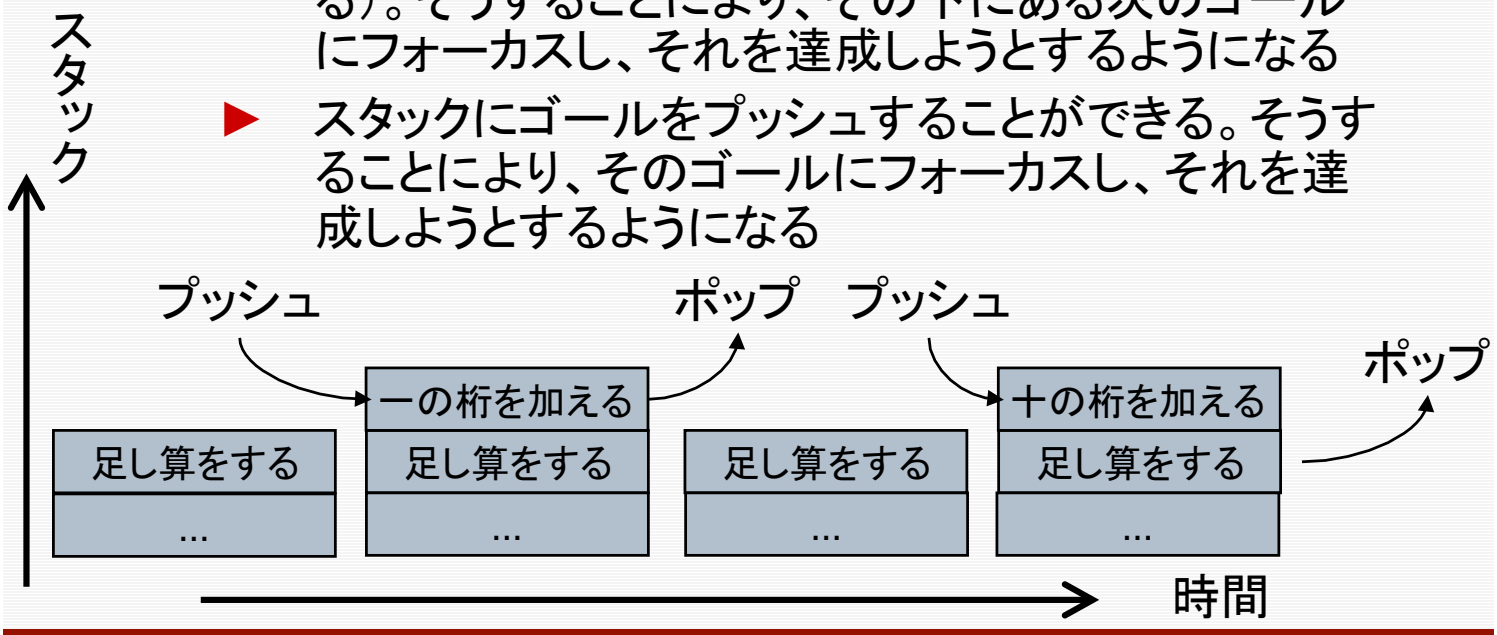
P9:最後の桁を処理する（繰り上げあり）
IF ゴールは「ある桁の数字を加える」、かつ、下段の数字が読み込み済み、かつ、それが＋である、かつ、繰上げ数字がある
THEN 「問題終了」を記銘、そして、繰上げ数字を書く、そして、ゴールをポップする

P10:次の桁に進む
IF ゴールは「足し算を行う」、かつ、処理すべき桁が決まっている、かつ、繰上げ数字が決まっている
THEN サブゴール「その繰上げ数字をつけて、その桁の数字を加える」を設定する、そして、次に加えるべき桁を記銘する

P11:終了する
IF ゴールは「足し算を行う」、かつ、「問題終了」が記銘されている
THEN ゴールをポップすることにより、問題を終了する

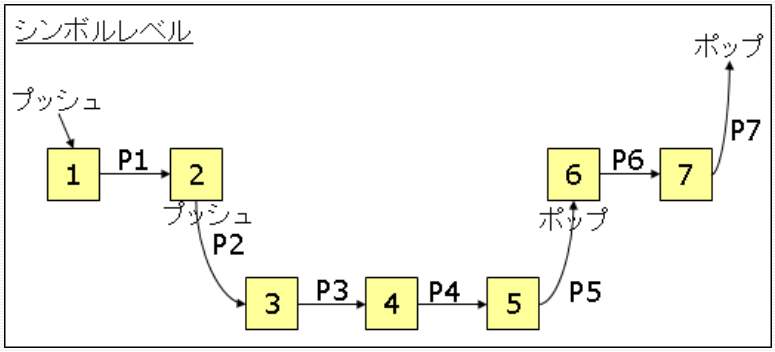
プロダクションルールのサイズと発火の制御

- プロダクションルールは、認知の基本ステップ
  - 外部から観測可能なサイズ
    - 発話、キーストローク、眼球運動など
  - 個々のルールが学習可能
    - 例：P6.繰り上がりを処理する
- プロダクションルールの発火の制御には、ゴール構造が重要
  - 目標は、ゴールスタックに符号化される
  - スタックの最上位のゴールを達成しようとする
  - 最上位のゴールが達成されれば、スタックからそのゴールをポップすることができる（スタックからなくなる）。そうすることにより、その下にある次のゴールにフォーカスし、それを達成しようとするようになる
  - スタックにゴールをプッシュすることができる。そうすることにより、そのゴールにフォーカスし、それを達成しようとするようになる

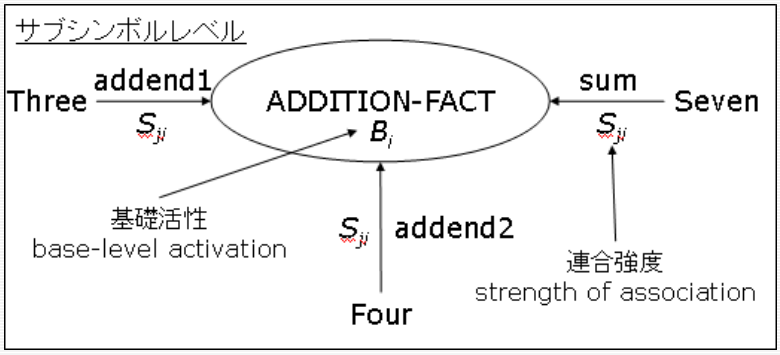


ACT-R の2つのレベル

1. シンボルレベル
   - ゴール状態とゴール状態を変更させるプロダクションに着目するレベル ‥ 足し算の例



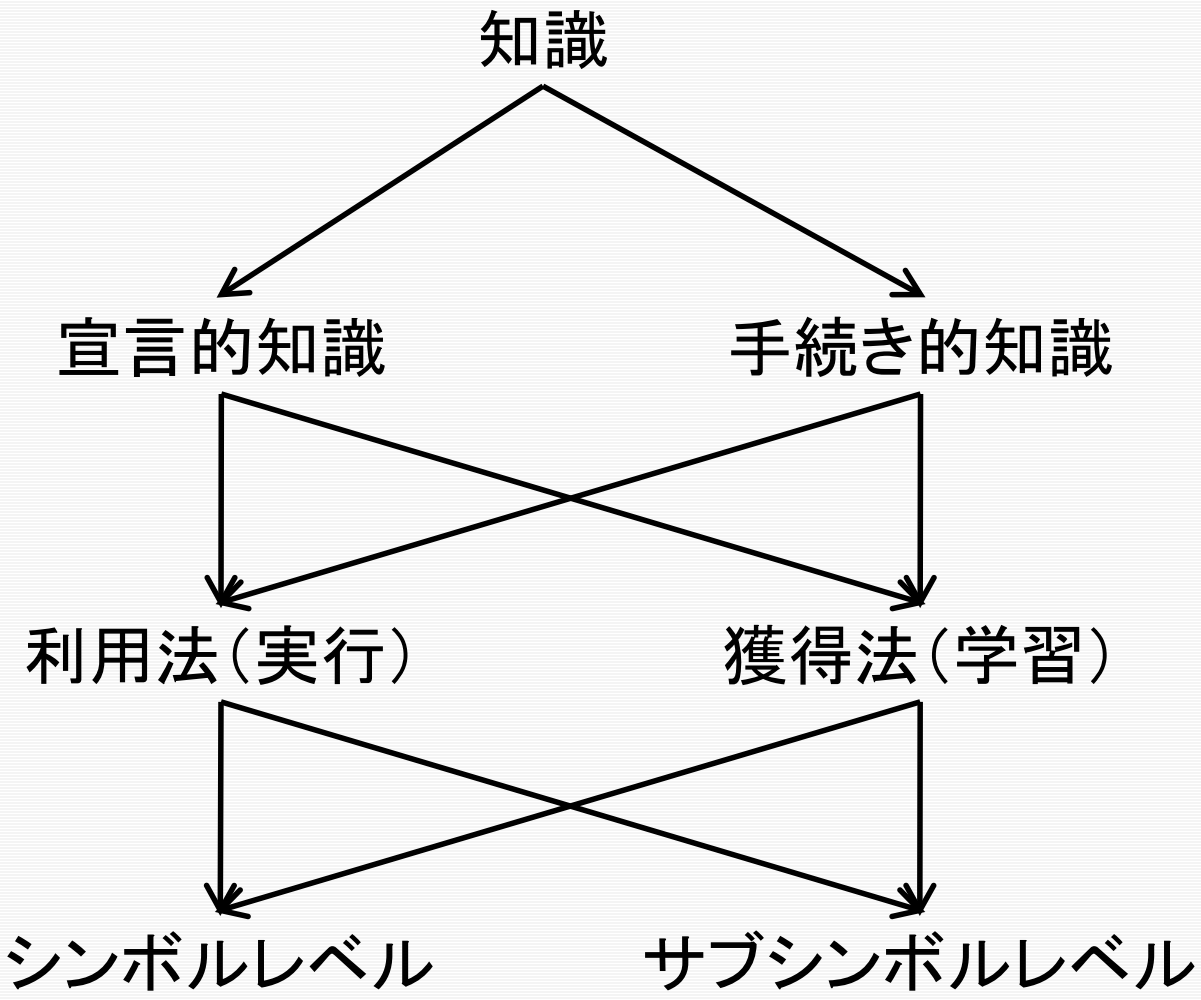
2. サブシンボルレベル
   - 基礎活性や連合強度が関与するレベル
     - 検索速度を決定する ‥ 神経回路に似た活性化プロセスにより宣言的知識が活性化
     - 学習により変化する → 検索速度や発火するプロダクションに影響



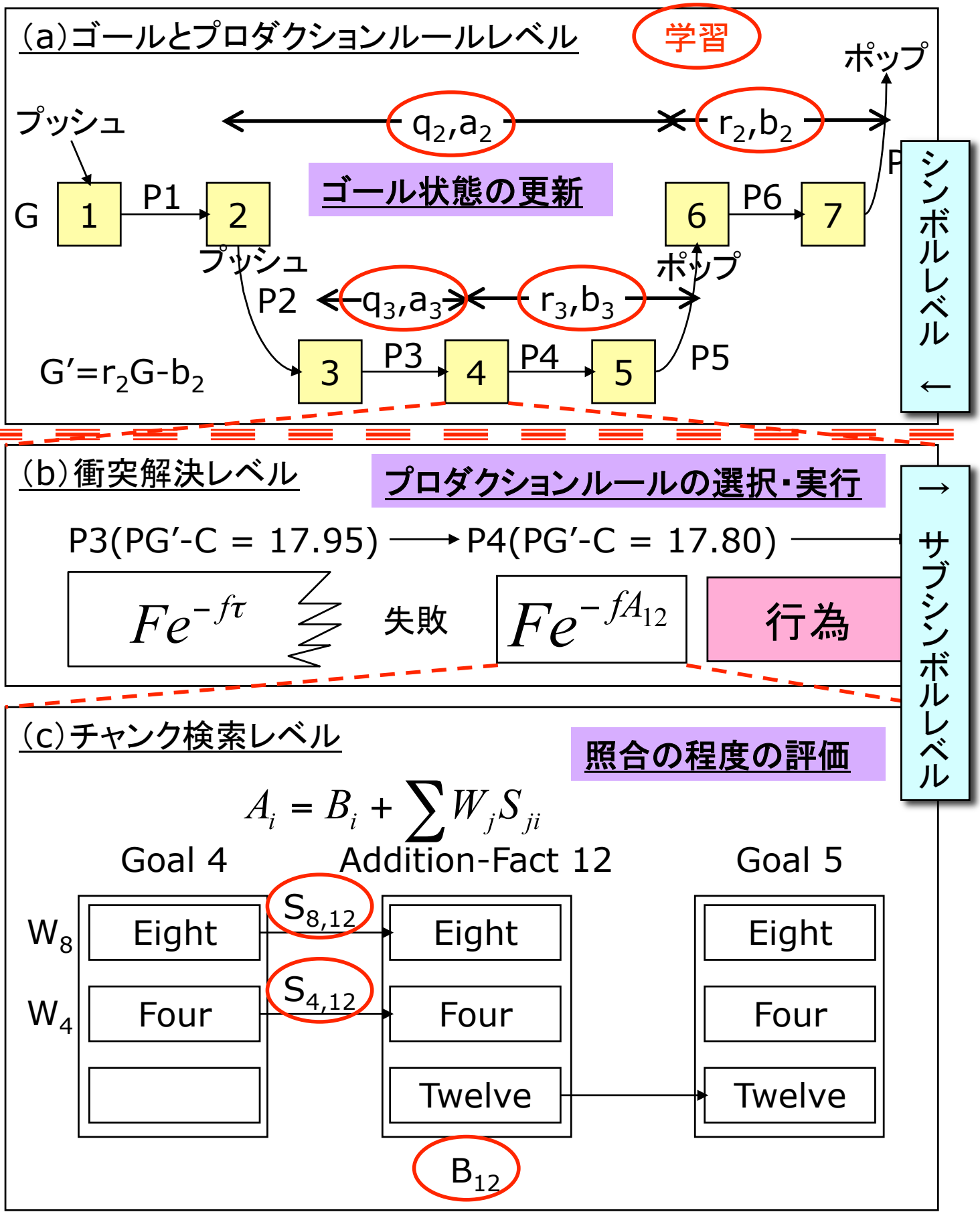
ACT-R の知識表現

1. 手続き的知識と宣言的知識の区別
2. 宣言的知識のチャンク表現
3. 手続き的知識のプロダクション表現
4. 行動の組織化におけるゴール構造の役割

ACT-R における知識の扱い

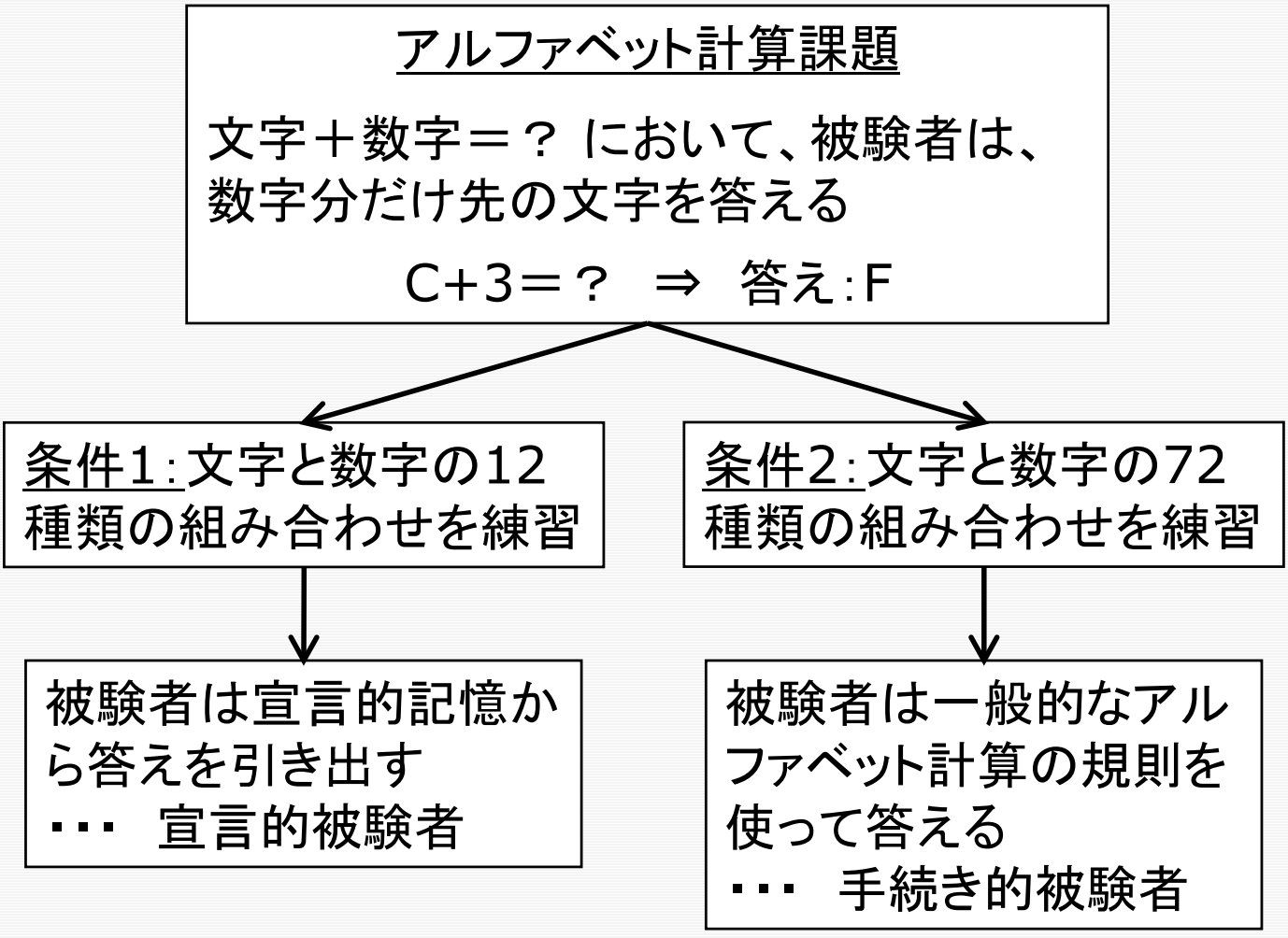


1．手続き的知識／宣言的知識の利用法（シンボルレベル）
2．手続き的知識／宣言的知識の利用法（サブシンボルレベル）
3．手続き的知識／宣言的知識の獲得法


(a)ゴールとプロダクションルールレベル
学習
ポップ
プッシュ
q_2,a_2
r_2,b_2
ゴール状態の更新
G
1
P1
2
6
P6
7
プッシュ
ポップ
P2
q_3,a_3
r_3,b_3
$G'=r_2G-b_2$
3
P3
4
P4
5
P5
シンボルレベル ←
(b)衝突解決レベル
プロダクションルールの選択・実行
P3(PG'-C = 17.95) → P4(PG'-C = 17.80)
$Fe^{-f\tau}$
失敗
$Fe^{-fA_{12}}$
行為
→ サブシンボルレベル
(c)チャンク検索レベル
照合の程度の評価
$A_i = B_i + \sum W_j S_{ji}$
Goal 4
Addition-Fact 12
Goal 5
W_8
Eight
$S_{8,12}$
Eight
Eight
W_4
Four
$S_{4,12}$
Four
Four
Twelve
Twelve
B_{12}

手続き的知識と宣言的知識の区別を示す実験

Rabinowitz and Goldberg (1995) の実験



Rabinowitz, M., & Goldberg, N. (1995). Evaluating the structure-process hypothesis. In F. E. Weinert & W. Schneider (Eds.), Memory Performance and Competencies: Issues in Growth and Development (pp. 225-242). Hillsdale, NJ: Lawrence Erlbaum.

手続き的知識と宣言的知識の違い ・・・ ACT-R による説明

被験者は宣言的記憶から答えを引き出す
・・・ 宣言的被験者

被験者は一般的なアルファベット計算の規則を使って答える
・・・ 手続き的被験者

問題開始時に適用されるプロダクション

P_n:アルファベット計算を行う
IF ゴールは「文字＋数字＝？に答える」
THEN サブゴール「数字分だけアルファベット文字を進める」を設定する、そして、答えを報告する

- *数字が大きくなると、解答時間が長くなる*

練習により獲得されるプロダクション

P_d:文字を答える
IF ゴールは「文字＋数字＝？に答える」、かつ、「文字、＋、数字、＝、別の文字」が記憶にある
THEN 別の文字を答えとして報告

「数字分だけアルファベット文字を進める」を実行するプロダクションが学習される

- *数字によらず、解答時間が一定*
- *P_nの学習は停止*

実験結果：
- 解答時間：宣言＜手続き
- 数字の影響：宣言＜手続き
- 新問題の解答時間：宣言＞手続き

手続き的知識と宣言的知識の違い　・・・　方向性

アルファベット計算課題（転移課題）

文字ー数字＝？ において、被験者は、数字分だけ前の文字を答える

F－3＝？　⇒　答え：C

ただし、C+3＝Fは学習済み

結果

知識転移の大きさ：

宣言的被験者＞手続き被験者

結論

- 足し算問題で獲得した宣言的知識を、引き算問題でも利用できる
  - ▶（＋,C,3,F）は（ー,F,3,?）の利用を促進する
- しかし、足し算問題で練習をした手続き的知識は、引き算問題には役立たない
  - ▶学習された手続き「（C,0）→（D,1）→（E,2）→（F,3）」は、逆向きの手続き「（F,3）→（E,2）→（D,1）→（C,0）」を促進しない
- 手続き的知識には方向性がある

チャンク（chunk）の構造

チャンクの定義：
- チャンクは、互いに独立した断片的な情報を、一組のスロット（slot）とそこに設定された値（value）の組み合わせによって符号化する
- 情報の種類は、isa スロットに指示される
- isa スロット以外のスロットは、それら全てが一体となって形成されるべき情報のパターンを表現する

＊スロット《テレビ番組などの時間枠》と同じ意味

Fact9+5	
isa	二つの数字の和
addend1	9
addend2	5
sum	14

数字1	
isa	視覚対象物
value	9
row	上段
column	1の位

２ ３ ９
＋ １ ２ ５

チャンクの2つの起源

1. ゴールの符号化 ・・・ ゴールチャンク
   - Fact9+5 は、「9と5の和がいくつになるかを見つける」というゴールが達成されたときの記録としてチャンクが生成される

Fact9+5	
isa	二つの数字の和
addend1	9
addend2	5
sum	14

　　２　３　９
＋　１　２　５

2. 環境中のオブジェクトの符号化 ・・・ オブジェクトチャンク
   - 一の位の上段の数字を読み込んだときに生成される

数字1	
isa	視覚対象物
value	9
row	上段
column	1の位

チャンクの起源とスロット構造の関係

1. チャンクがゴールに起源を持つ場合
   - isa には、スロット間の関連を記述
     - 二つの数字の和
   - スロット値に引数を指定
     - 9, 5, 14

2. チャンクが知覚されたオブジェクトに起源を持つ場合
   - 対象物の型
     - 視覚対象物
   - 上位の構造から見たその対象物の位置
     - 上段 ・・・ 足すべき数字の一方
   - その対象物の置かれている場所
     - 一の位

- ゴールチャンクとオブジェクトチャンクは異なった種類のスロットを持つが、シンタックスは同一
- プロダクションシステムにおいては、区別されず、同じように処理される

プロダクションルールの概要

- プロダクションルールは、認知のステップに対応
- プロダクションの基本構造：

ゴール条件テスト ＋ チャンク検索 → ゴール状態の変換

- ゴール条件テスト：ゴール状態に関するテスト。テストに通れば、プロダクションが選択される。そして、チャンク検索が実行される。複数のプロダクションがゴール条件テストを通過したときは、衝突解決（conflict resolution）プロセスが適用され、一つが選択される
- チャンク検索：チャンクパターンを宣言的知識と照合し、情報を検索する。
- ゴール状態の変換：現在のゴールの記述と、検索された情報に基づいて、ゴール状態を変換する。変換の仕方には、以下の場合がある：
  - 現在のゴールの状態を更新する
    - 同じプロダクションが繰り返し実行されることを回避
  - 現在のゴールをポップする
  - サブゴールを生成しプッシュする
    - サブゴールは、さらに別のプロダクションを実行するか、運動プログラム（手の動き、眼の動き、など）を実行することによって達成される
    - したがって、サブゴールを生成し、プッシュすることが、実質的に、行為を実行することに対応

ACT-R の「認知」のイメージ

- いくつかのプロダクションが発火し数百ミリ秒ごとにゴール状態を変える（潜時《刺激と反応間の時間》は、50ミリ秒～1秒）
- 思考は、ゴールを変換することにより、逐次的に進展する
- ゴール変換を行う為に、サブシンボルレベルでは、膨大な量の並列処理が行われている

プロダクションルールの特徴

1. モジュール性（modularity）
   - プロダクションルールは、手続き的知識が獲得され利用されるときの基本単位
   - 複雑なスキルの学習（例えば、LISP関数の学習）において学習されるものは、個々のプロダクションルールに対応
2. 抽象性（abstraction）
   - プロダクションルールは、刺激-応答結合（stimulus-response bond、例：パブロフの犬）とは違っている
   - ひとつのプロダクションルールが、様々な状況で適用され異なった行為を生じさせる → 汎用性
   - 「P4：数字を加える」は任意の2数字に適用できる。変数の導入により実現

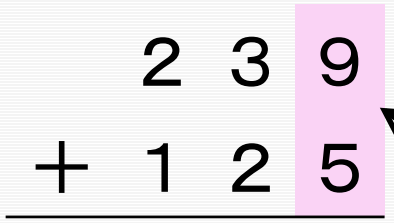
P4：数字を加える
IF ゴールは「ある桁の数字を加える」、かつ、
上段の数字と下段の数字が符号化されていて、和が未定、かつ、和を記憶から検索できる
THEN 和を記銘する

プロダクションルールの特徴

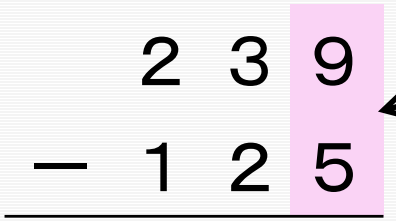
3. ゴール分解性（goal factoring）
   - プロダクションルールの適用範囲を規定する
   - 抽象性によって実現された汎用性を制約する
   - プロダクションルールが、特定のゴールタイプに限定して適用される
     - ゴールが「和を計算する」場合に適用されるゴール群と、「差を計算する」場合に適用されるゴール群が異なっている
   - 同じ外部環境に対して、異なった振る舞いを生じさせる
   - 刺激-応答結合では難しい ・・・ 刺激の表現が貧弱で状況適応が難しい
4. 条件・行為の非対称性（conditional asymmetry）
   - 刺激-応答結合と類似
   - 他の認知理論とは異なっている
     - パタン完成型神経システム、prologシステム
   - 宣言的知識と、大きく異なっている
     - 4×3＝12は、4,3 → 12、4,12 → 3 で利用可
     - 桁数の大きい掛け算をするのに必要なプロダクションを学習しても、それを一般化して桁数の大きい割り算に利用できるようにはできない
   - 要するに、柔軟性を捨て、アクセス効率を高めた

同じ外部環境に対して、異なった振る舞いを生じさせる

P4-A:数字を加える
IF ゴールは「ある桁の数字を加える」、かつ、上段の数字と下段の数字が符号化されていて、和が未定、かつ、和を記憶から検索できる
THEN 和を記銘する（記憶から検索）



ゴールの適用範囲を制限
・・・ ゴール分解性



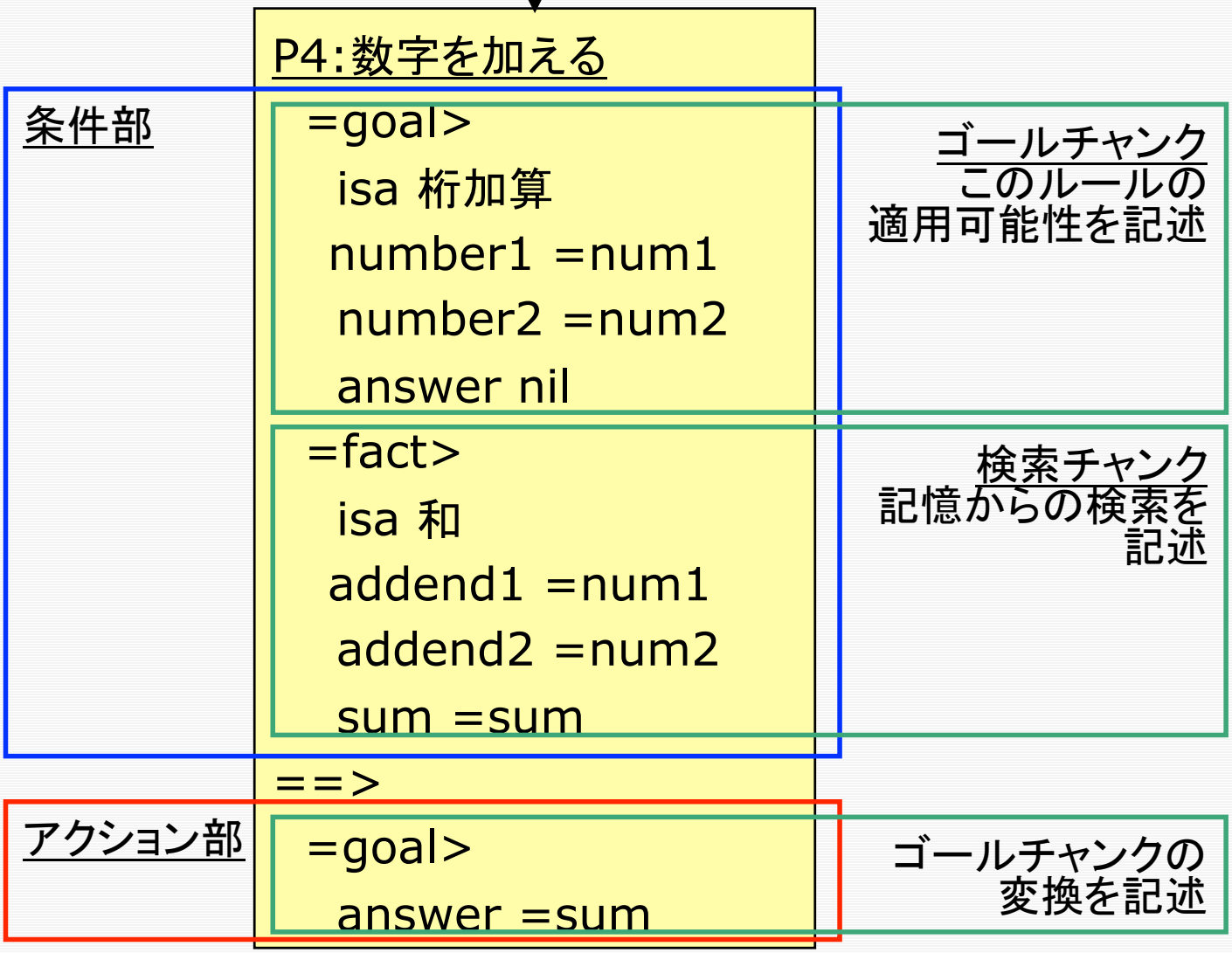
1の位の数字の
並びは同じ

P4-S:数字を引く
IF ゴールは「ある桁の数字を引く」、かつ、上段の数字と下段の数字が符号化されていて、差が未定、かつ、差を記憶から検索できる
THEN 差を記銘する（記憶から検索）

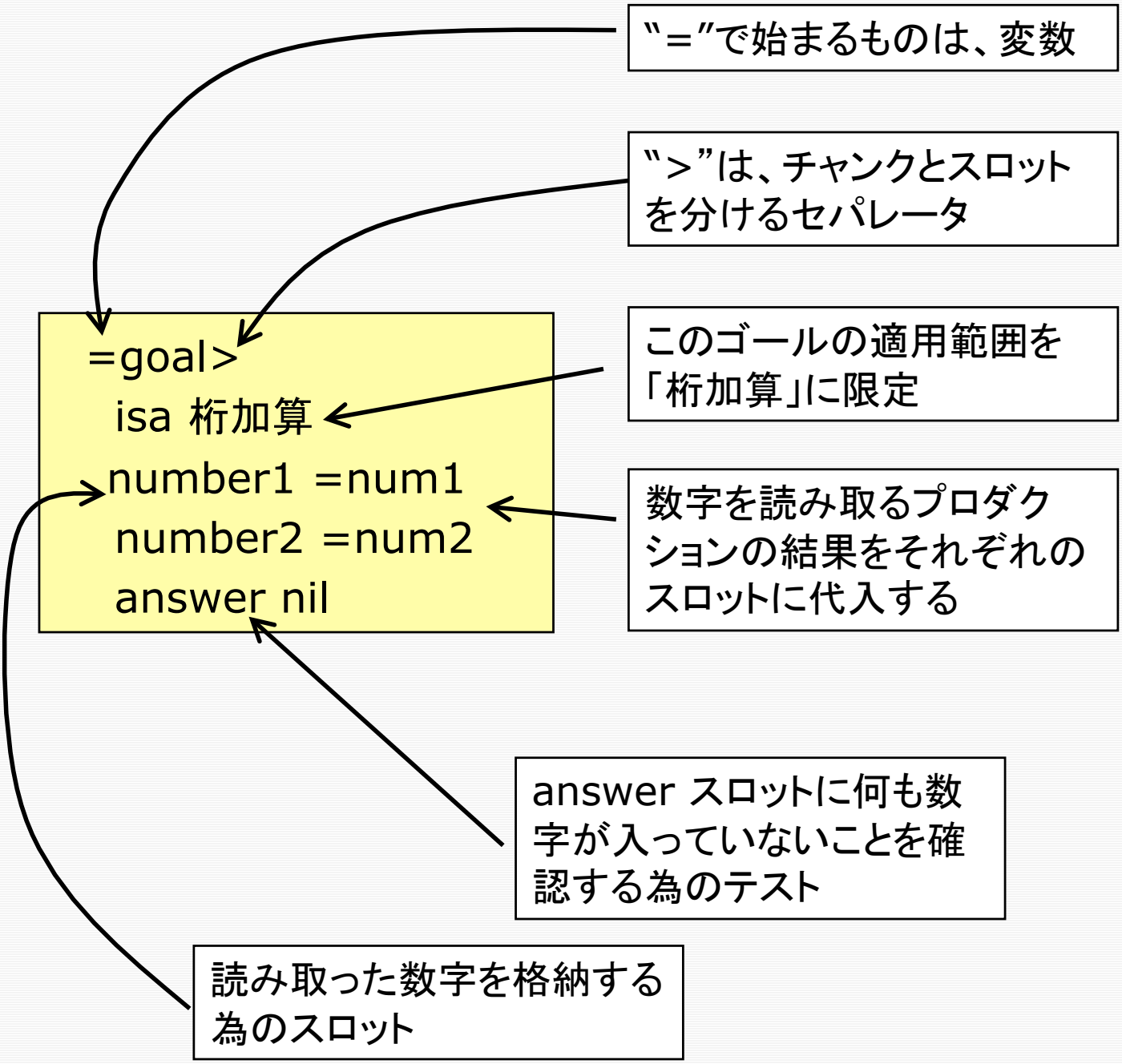
プロダクションのシンタックス

- プロダクションは、ゴール変換を詳細に記述する

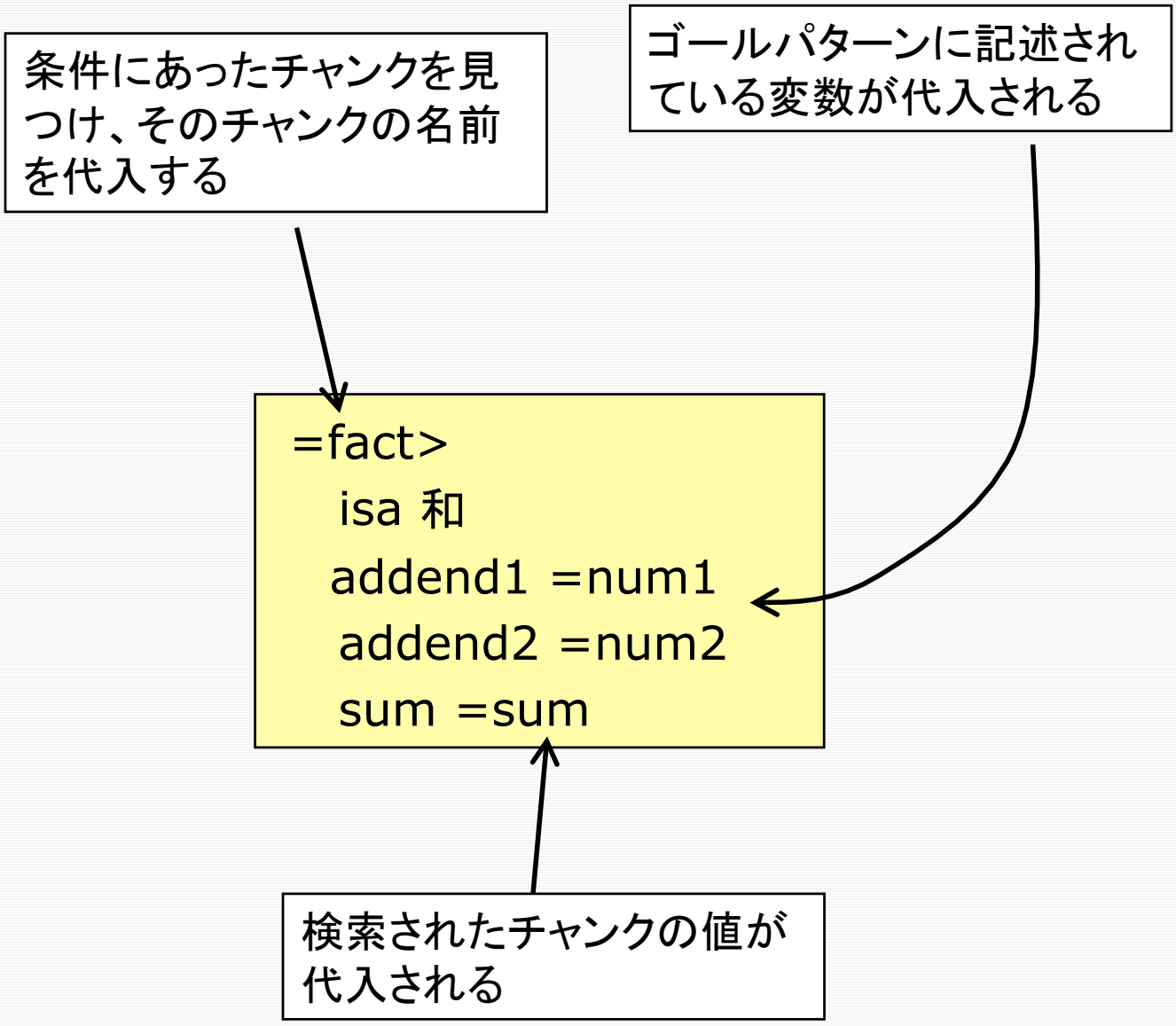
P4:数字を加える
IF ゴールは「ある桁の数字を加える」、かつ、上段の数字と下段の数字が符号化されていて、和が未定、かつ、和を記憶から検索できる
THEN 和を記銘する



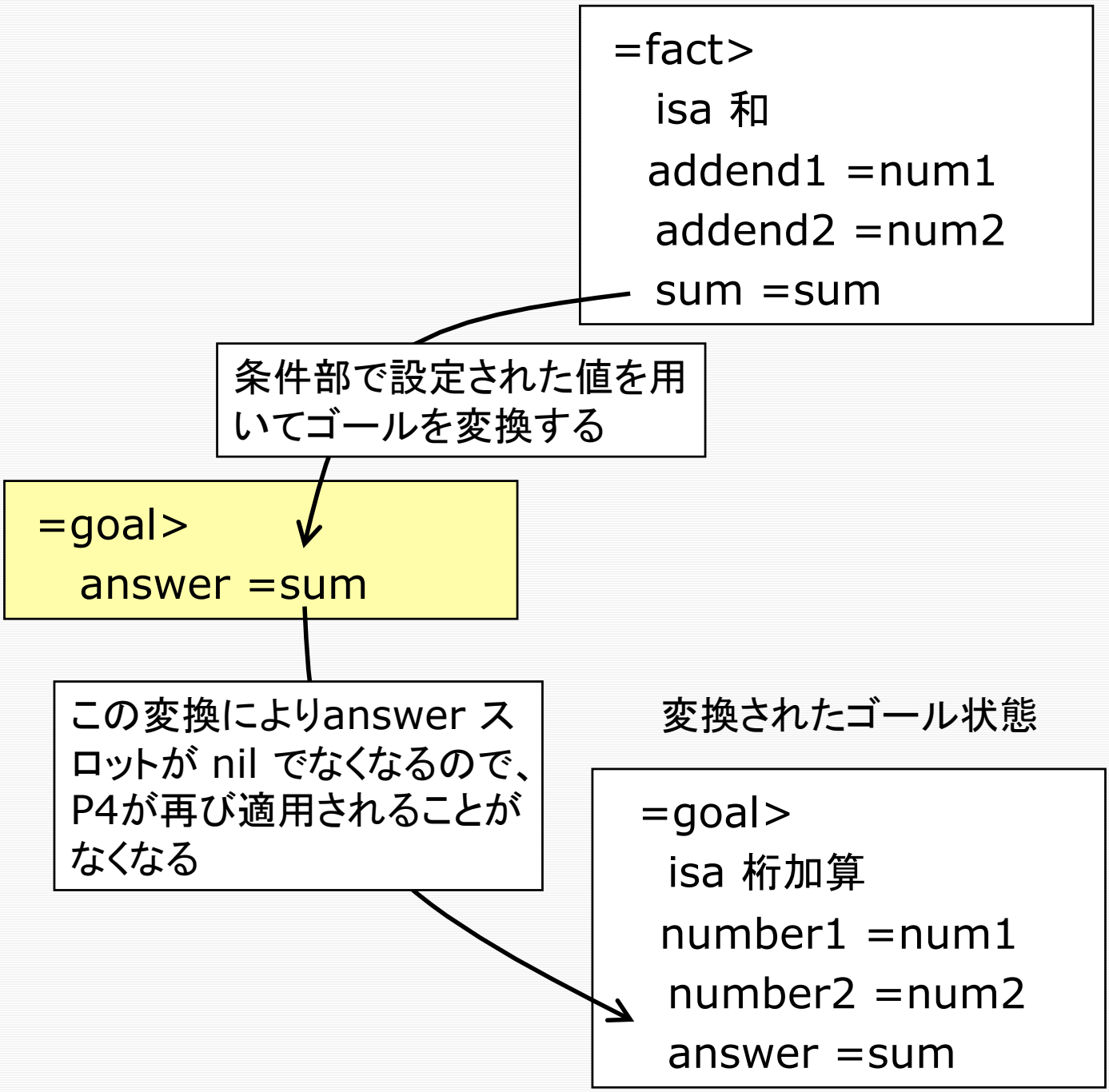
ゴールチャンク


“=”で始まるものは、変数
“>”は、チャンクとスロットを分けるセパレータ
=goal>
isa 桁加算
number1 =num1
number2 =num2
answer nil
このゴールの適用範囲を「桁加算」に限定
数字を読み取るプロダクションの結果をそれぞれのスロットに代入する
answer スロットに何も数字が入っていないことを確認する為のテスト
読み取った数字を格納する為のスロット

検索チャンク


条件にあったチャンクを見つけ、そのチャンクの名前を代入する
ゴールパターンに記述されている変数が代入される
=fact>
isa 和
addend1 =num1
addend2 =num2
sum =sum
検索されたチャンクの値が代入される

アクション部:ゴール変換



2桁2数足し算のプロダクションルール（プログラム版）

P1:開始する
```
 =goal>
  isa 足し算
  column nil
==>
 =newgoal>
  isa 桁加算
  column 一の桁
  note =carry
  carry ゼロ
 =goal>
  column 十の桁
  carry =carry
 !push! =newgoal
```

P2:上段の数字を読む
```
 =goal>
  isa 桁加算
  number1 nil
  column =col
 =object>
  isa 視覚対象物
  value =num1
  row 上段
  column =col
==>
 =goal>
  number1 =num1
```

P3:下段の数字を読む
```
 =goal>
  isa 桁加算
  number2 nil
  column =col
 =object>
  isa 視覚対象物
  value =num2
  row 下段
  column =col
==>
 =goal>
  number2 =num2
```

P4:数字を加える
```
 =goal>
  isa 桁加算
  number1 =num1
  number2 =num2
  answer nil
 =fact>
  isa 和
  addend1 =num1
  addend2 =num2
  sum =sum
==>
 =goal>
  answer =sum
```

P5:答えを出す
```
 =goal>
  isa 桁加算
  answer =sum
  carry ゼロ
 =sum>
  isa 数字
  tens =num
  ones =number1
==>
 =goal>
  answer =number1
  note =num
```

P6:繰り上がりを処理する
```
 =goal>
  isa 桁加算
  answer =number
  carry イチ
 =fact>
  isa 和
  addend2 イチ
  addend1 =number
  sum =new
==>
 =goal>
  answer =new
  carry ゼロ
```

P7:答えを書く
```
 =goal>
  isa 桁加算
  carry ゼロ
  answer =number
  column =col
 =number>
  isa 数字
  ones =value
  tens ゼロ
==>
 !output! =value
 !pop!
```

P8:最後の桁を処理する（繰り上げなし）
```
 =goal>
  isa 桁加算
  number2 +
 -carry イチ
==>
 =goal>
  note 終了
 !pop!
```

P9:最後の桁を処理する（繰り上げあり）
```
 =goal>
  isa 桁加算
  number2 +
  carry イチ
  column =col
  carry =carrynum
 =carrynum>
  isa 数字
  ones =value
==>
 !output! =value
 =goal>
  note 終了
 !pop!
```

P10:次の桁に進む
```
 =goal>
  isa 足し算
  column =pos1
 -carry 終了
 =next>
  isa 次の情報
  before =pos1
  after =pos2
==>
 =newgoal>
  isa 桁加算
  column =pos1
  carry =new
 =goal>
  column =pos2
  carry =newcarry
 !push! =newgoal
```

P11:終了する
```
 =goal>
  isa 足し算
  carry 終了
==>
 !pop!
```

ゴール構造（goal structure）

- ゴール構造は、プロダクションシステムの重要な構成要素
  - ACT-R, Soar, EPIC, 3CAPS
  - 過去の研究の教訓：
    - 人間の認知を適切に表現するには、各時点での行動目的を表現し、その目的に応じて行動を系統立てることが必要である
    - 認知システムの構成要素にゴール構造が存在しないと、系統立てられた認知行動を達成することができない
- ゴール構造は、認知に連続性を生じさせる
  - ゴール構造の更新により、新しいゴール状態（目的）が生成される
  - ゴール状態の系列として認知を捉えることもできる
  - ただし、このことが、必ずしもプロダクションルールの系列的発火を意味するわけではない。各サイクルで、複数のプロダクションが発火し、いくつかのサイクル後にゴール状態が変わるというプロダクションシステムもある（EPIC, Soar, 3CAPS）

ACT-R のゴール構造

- 特徴
  - ゴールはスタック状に積まれる
  - 最上位のゴールが現在のゴールになる
  - プロダクションルールは、そのゴールを変更したり、ポップしたり、他のゴールをスタックにプッシュする
- ゴール構造（スタック）の例
  1. 3に4を加える
  2. X-4＝3 を解く
  3. スクリーンに表示された問題を解く
  4. 実験の単位を取る
  5. 心理学基礎を修了する
  6. 学士号をとる
  - ゴール構造の上のほうだけを扱う

スタック状にゴールを積むことに起因する問題とACT-Rの対処

- 問題：割り込み
- 対処：

IF　ゴールは「ある課題を行う」、
　　かつ、「火事だ」という声を聞く
THEN　「避難する」にゴールを変更する

- 問題：複数のゴールの並列実行
- 対処：
  - 複数ゴールタスクを実行するという単一のゴールを設定する
  - そして、複数ゴールの個々のゴールに対応するサブゴールを交互に系列的に実行する、あるいは、複数ゴールタスクを単一のものとして実行する
- 問題：ゴールスタックを完全に覚えているという仮定
- 対処：今後の課題

プロダクションルールのタイプ

プロダクションルールは、ゴール変換の方法を決定するための手段

ゴール変換＝ゴールスタック×現在のゴール

		ゴールスタック		
		プッシュ	ポップ	何もしない
現在のゴール	変更する	4 結果を返す	6 変化ありポップ	2 ゴールの精緻化
	何もしない	3 副作用	5 変化なしポップ	1 変化なし

1．変化なし

- 環境への行為を含む
- 無限に繰り返されないように、環境に関するテストを含む

```
槌で打つ
  =goal>
    isa 槌で釘を打つ
   object =釘
  =state>
    isa 釘の状態
   object =釘
    state 頭が出ている
==>
    execute 槌で打つ
```

2. ゴールの精緻化

- 現在のゴールをより精緻なものにする

P2:上段の数字を読む
=goal>
isa 桁加算
number1 nil
column =col
=object>
isa 視覚対象物
value =num1
row 上段
column =col
==>
=goal>
number1 =num1

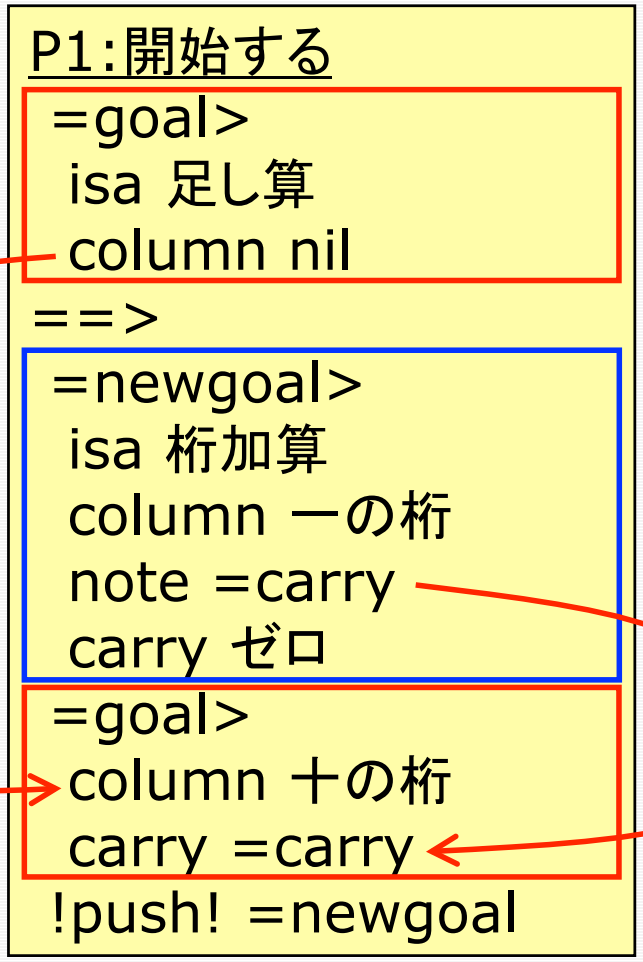
3. 副作用

- 現在のゴールを変更せずに、新しいゴールがプッシュされる

```
教師に質問する
 =goal>
  isa 方程式を解く
  equation =3次方程式
  factor1 nil
  fcator2 nil
  fcator3 nil
=equation>
  isa 方程式
  Xの3乗の係数 1
  Xの2乗の係数 -6
  Xの1乗の係数 11
  Xの0乗の係数 -6
==>
 =newgoal>
  isa ヘルプを求める
  object =equation
  resource 教師
  !push! =newgoal
```

4. 結果を返す

- 新しいゴールをプッシュすると同時に、現在のゴールを精緻化する。プッシュされたゴールが完了したときに、同じプロダクションが再び発火しないようにする。



サブゴールが達成されたら、次は十の桁

サブゴールリターンメカニズム：
サブゴールの note スロットの変数 =carry は、サブゴール達成時には値が定まる。その値が、親ゴールの carry スロットに渡される

5. 変化なしポップ

P7:答えを書く
=goal>
isa 桁加算
carry ゼロ
answer =number
column =col
=number>
isa 数字
ones =value
tens ゼロ
==>
!output! =value
!pop!

例えば、=number に EIGHT が入る

EIGHT
isa 数字
ones 8
tens ゼロ

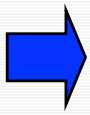
「8」が書き出される

6. 変化ありポップ

6+3=?

数えるのを止める
=goal>
isa 合計演算
arg1 =数値1
arg2 =数値2
count =数値2
current =答え
==>
=goal>
sum =答え
!pop!

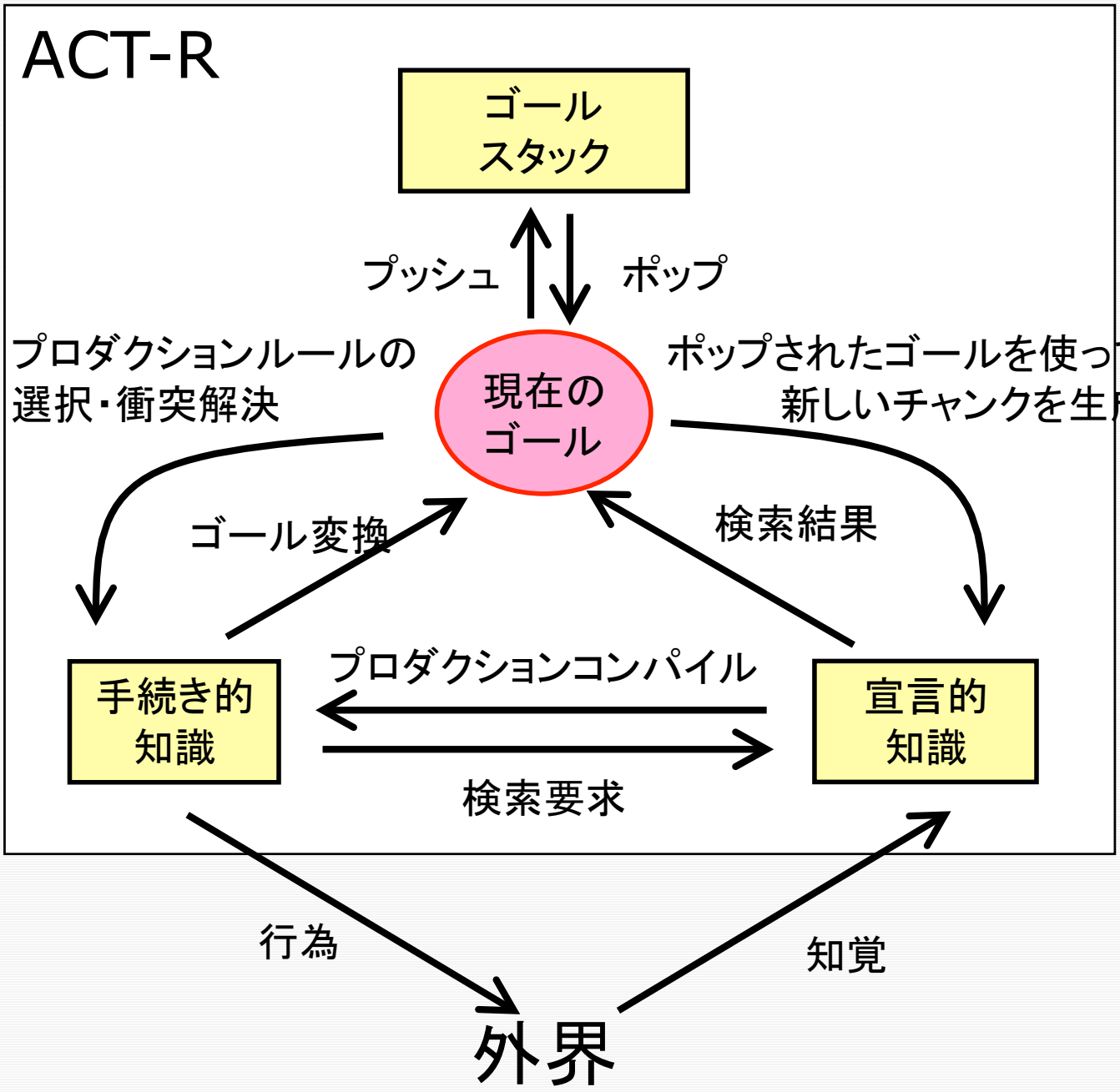
	0	1	2	3
arg1	6	6	6	6
arg2	3	3	3	3
count	0	1	2	3
current	6	7	8	9
sum	-	-	-	9



6+3>
isa 合計演算
arg1 6
arg2 3
count 3
current 9
sum 9

ポップされたゴールは宣言的知識として記憶される。将来、6+3=? に遭遇したとき、検索され、問題を解くのに利用される

ACT-R における情報の流れ



ACT-R における情報の流れ（説明）

- 3種類の記憶がある
  - ゴールスタック、手続き的知識、宣言的知識

- 現在のゴールがこれらを有機的に系統立てる
  - ゴールスタック
    - ゴールをスタックにプッシュする
    - ゴールをスタックからポップする
  - 手続き的知識
    - 現在のゴールに適合するプロダクションルールを手続き的知識の中から選択する
    - 複数のプロダクションルールが条件を満たすときに衝突の解決をする
    - 現在のゴールを変換する
      - ✓ 宣言的知識に対して検索要求を行う
      - ✓ 外界に対して行為を行う
  - 宣言的知識
    - ポップしたゴールから新しいチャンクを生成する
    - 現在のゴールに検索結果を返す
      - ✓ 外界を知覚し新しいチャンクを生成する
      - ✓ 新しいプロダクションを生成する（プロダクションコンパイル）

プロダクションルールは、宣言的知識の検索の順序付けをしている

- 個々のプロダクションは、通常1つ、多くとも2-3個の宣言的チャンクを検索する
- 検索されたチャンクの内容に基づいて、いくつかの可能な行為のなかから一つを選択するという選択的な決定を行うことはできない
- むしろ、選択的な決定は、検索された情報をゴールチャンクに置き、さらに情報検索を進め、その後のサイクルで、あるプロダクションを発火させ、他のものを発火させないという形で、行われる

- *要するに、少しずつ情報を集め、進む道を微調整しながら、最終決定をするのであり、複雑な条件を設定し、大量の情報を収集し、一気に決定をするのではない*