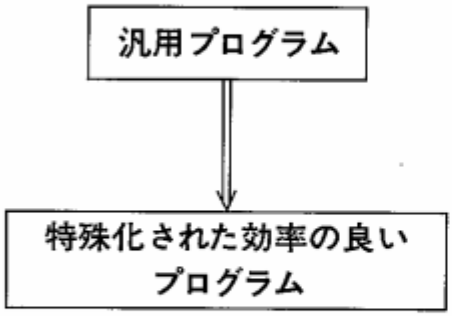
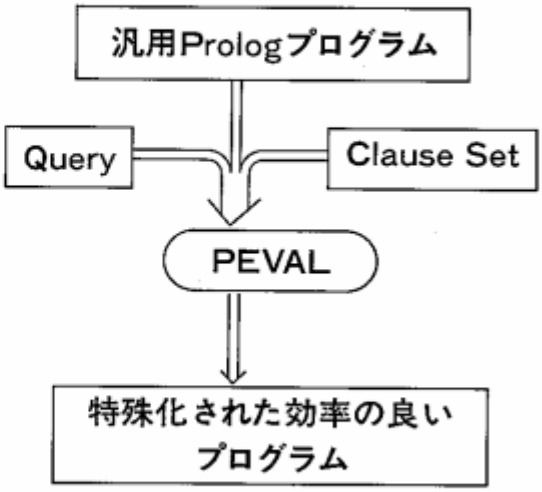
メタ推論と部分計算

PEVAL：a Partial EVALuator for Prolog Programs

部分計算



Prologプログラムの部分計算



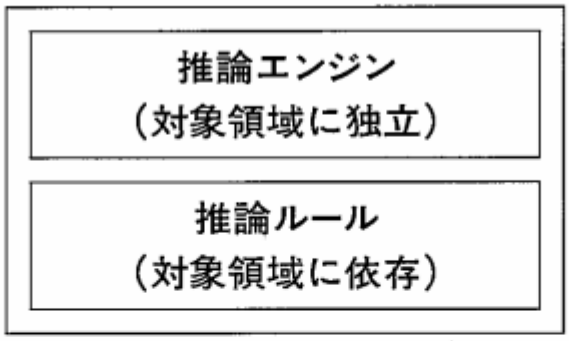
論理型推論システム

『Prologはシステム構築用の良い言語である。』

理由１：ユニフィケーション
　　パターン・マッチングより強力

理由２：バックトラッキングによる探索
　　推論システムにとっては本質的

推論システムの概念構成



推論エンジン
　推論の制御
　推論の観察
　推論過程の説明

推論ルール
　対象領域に特有の知識

どうやってPrologで論理型推論システムを構築するか？

(1) 推論エンジンをルール・インタプリタとして実現する。

メタ・プログラミング　アプローチ

理解し易い，管理し易い，修正し易い
効率が悪い

(2) Prolog処理系を推論エンジンとして用いる。

トランスレータ　アプローチ

効率が良い
理解しにくい，管理しにくい，修正しにくい

メタ・プログラミング

推論についての高階の記述を可能にする。

強力な概念を自然に導入することができる。

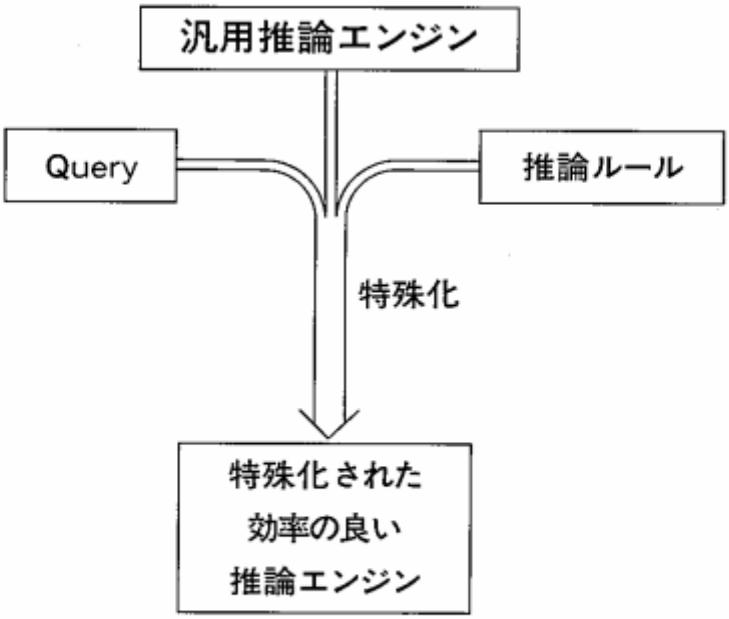
推論のメタ・レベルの制御

推論過程の観察及び設明

問題： 2 つのアプローチの融合

汎用性と効率の間のトレード・オフの解消

解答：部分計算がこのトレード・オフを解消する



部分計算を基本ツールとする推論システムの構築法

ステップ1

推論エンジンと推論ルールを明確に分離して構成

{メタ・プログラミングの表現力の活用}

ステップ2

汎用の推論エンジンを特定の推論ルールに関して部分計算

ステップ3

特殊化されたプログラムを直接実行

{効率の良い推論}

適用実験

(1) Certainty factorを扱う推論システム

平均3倍の高速化

(2) ボトム・アップ・パーザ

平均5倍の高速化

BUPトランスレータと同等の効率を実現

(3) 数式処理システム

平均5倍の高速化

メタ・インタプリタの段階的特殊化

段階的に生成されるオブジェクト・プログラムに関するメタ・インタプリタの段階的特殊化

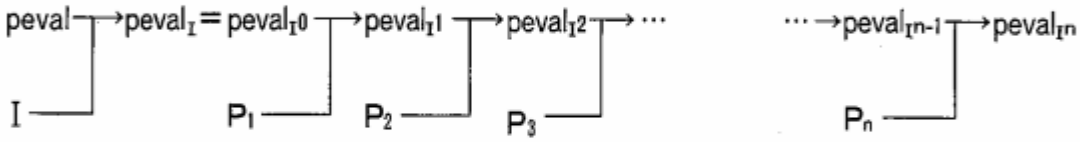
推論システムにおいては、推論ルールが段階的に得られるときに有効

例　推論ルールの段階的獲得，学習

オブジェクト・プログラム P，メタ・インタプリタ I

$$P = \{P_1, \ldots, P_n\}$$

$$\{P_1\} \rightarrow \{P_1, P_2\} \rightarrow \{P_1, P_2, P_3\} \rightarrow \cdots \rightarrow \{P_1, \ldots, P_n\}$$



Compiler：$peval_I$

Partially Specialized Compiler：$peval_{Ik}$