ICOT Technical Report: TR-032

TR-032

逐次型Prologプログラム の静的解析

尾内 理紀夫、益田 嘉直、麻生 盛敏

December, 1983

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

1．はじめに

DEC-10 Prolog [18]program のstatic analyzer を開発し，ＩＣＯＴで開発された各種プログラムを静的に解析し，各種data

たとえば，OR Relation 数，AND literal 数，predicate の参照数，head predicate の引数個数，head predicateの構造data引数個数，body側の引数個数，body側構造data引数個数，組込み述語使用頻度，CUT ratio ，等

を収集した。このstatic analyzer はDEC-10 Prolog によって記述され，clause数は，約230である。この解析の結果，

①DEC-10 Prolog は逐次実行型であり，メモリ空間が十分でない。これに起因して，プログラムの多くは，cut を多用しており，deterministic なプログラムとなっている。

②AND literal の約半分は組込み述語であり，programmerはDEC-10 Prolog の多種多様な組込み述語を巧みに使っている。また組込み述語のうち約30% はI/O であり，I/O の速度がプログラム実行速度に影響を及ぼす。

③database clause のOR relation 数は inference clause のそれの約４倍であり，database clause が大規模化するとこの比は増大するであろう。よって大規模database clause を含むプログラムの場合，database clause に関する unification の高速化がPrologプログラム実行の高速化に大きな影響を及ぼす。

等の静的特性が考察された。本稿では，解析結果及びそれに基づく静的特性の考察について述べる。

2．収集data項目

本static analyzer はDEC-10 Prolog プログラムを先頭から読み，以下の各種dataを出力する。

2.1 OR relation数

同一head predicate symbol を持ち，引数個数が同一のclause同志をOR relation にあるという。そのようなclause同志の個数をOR relation 数と呼ぶ。たとえば，次のようなclauses があった時，inference clause（後述）（P.1)のOR relation 数は３である。

```
p([]).
p([X | Y]):-q(Y).
p(1):-r(1).
```

Total OR relationsの定義を次に示す。（Σはプログラム内のclauseについてのsummation ）

Σ OR relations

2.2 inference clause

OR relation にあるclauses の中に少なくとも一つのruleを含むclauseをinference clauseと呼び，（predicate name , arity) で示す。

inference clauseの平均OR relationsの定義を次に示す。

Total OR relations of inference clauses ／inference clauseの種類数

2.3 database clause

OR relation にあるclauses すべてがunit clausesのみから構成されるclauseをdatabase clause と呼び，（predicate name, arity)で示す。

このstatic analyzer では，cut ! は，AND literal には含めない。よって，たとえば，h :-!. は，ruleではなく， h. というunit clause とみなす。

database clause の平均OR relationsの定義を次に示す。

Total OR relations of database clauses／database clause の種類数

2.4 参照数

あるclauseが参照された回数，即ち，そのclauseのhead predicateがbody側に現れた回

数である。

inference clauseの平均参照数の定義を次に示す。（Σはプログラム内のinference clauseについてのsummation ）

（Σ inference clause への参照数）／inference clauseの種類数

2.5 head predicate の引数

head predicate引数の平均個数の定義を次に示す。（Σがプログラム内のinference clauseについてのsummation ならば，inference clauseのhead predicate引数の平均個数，Σがdatabase clause についてのsummation ならば，database clause のhead predicate引数の平均個数である。）

（Σ head predicate の引数個数）／Total OR relations

2.6 head predicate の構造data引数

head predicate引数の内var,atom,integer以外の引数個数である。（var,atom,integer以外の引数を構造data引数と呼ぶことにする。）
head predicateの構造data引数の平均個数の定義を次に示す。（Σはプログラム内のclauseについてのsummation ）

（Σ head predicate の構造data引数個数）／Total OR relations

2.7 AND literals

AND literal の定義を次に示す。

①body側の“，”で区切られたliteral の個数。

②但し，“：”は，“，”と同等とみなす。

③但し，組込み述語　is,>,>=,<,=<, 及び，＼＋に関しては，その引数の内部構造をも解析する。

例えば，

```
p(X,Y,Z):- X>Y,Z is Y-X.
```

のbody側は，三つのliteral　_>_，_is_ ,_ -_から構成されていると見なす。

一つのinference clause当りの平均AND literalsの定義を次に示す。（Σはプログラム内のinference clauseについてのsummation ）

(Σ AND literals) ／ Total OR relations of inference clauses

2.8 組込み述語(Evaluable predicate) 数

一つのinference clause当りの平均組込み述語数の定義を次に示す。(Σはinference clauseについてのsummation)

(Σ Evaluable predicates in an inference clause) ／ Total OR relations of inference clauses

2.9 Evaluable predicate ratio

Evaluable predicate ratio の定義を次に示す。(Σは，プログラム内のinference clauseについてのsummation)

(Σ Evaluable predicates in an inference clause) ／ (Σ AND literals)

2.10 body側引数

AND literal の引数の総和である。

1つのinference clauseあたりの body 側の平均引数個数の定義を次に示す。(Σは，プログラム内のinference clauseについてのsummation)

(Σ body 側引数個数) ／Total OR relations

2.11 body側構造data引数

AND literalsの構造data引数の総和である。

1つのinference cluase当りの構造 data 引数の平均個数の定義を次に示す。(Σは，プログラム内のinference clauseについてのsummation)

(Σ body 側構造 data 引数個数) ／Total OR rerations

2.12 cut ratio

cut ratio の定義を次に示す。(Σは，プログラム内のinference clauseについてのsummation)

(Σ cut numbers in an inference clause) ／ Total OR relations

3．静的解析結果と静的特性

ICOTで開発された各種DEC-10 Prolog プログラムの解析結果を示すとともに，それから考察されるDEC-10 Prolog プログラムの特性について述べる。

（各プログラムの機能概要については付録１参照，より詳しくは，参考文献参照）

3.1 inference clauseの解析結果と特性

(1) 解析結果

表１（次ページ）に，inference clauseに関する解析結果を示す。

縦項目

・プログラムのNoは，付録１のプログラム番号に対応している。

横項目（定義は２章を参照）

- rul : inference clauseの種類数
- OR : Total OR relations of inference clause
- av OR : 平均の OR relations
- m OR : 最大 OR relations
- av REF : 平均参照数
- m REF : 最大参照数
- av Harg : head predicateの引数の平均個数
- av H-SD : head predicateの構造 data 引数の平均個数
- H-SD/H : head predicate引数内に占める構造 data 引数の割合
- av AND : １つの inference clause 内の平均AND literal 数
- m AND : 最大 AND literal数
- av EV : １つのinference clause内の平均組込み述語数
- EV/AND : Evaluable predicate ratio
- av Barg : １つのinference clauseの body 側引数の平均個数
- av B-SD : １つのinference clauseの body 側構造 data 引数の平均個数
- B-SD/B : body 側引数内に占める構造 data 引数の割合
- av CUT : cut ratio

表１．　inference clause の解析結果

No.	Name	rui	OR	avOR	mOR	avREF	mREF	avHarg	avH-SD	H-SD/H	avAND	mAND	avEV	EV/AND	avBar	avB-SD	B-SD/B	avCUT
1	buptr	50	90	1.80	4	3.34	19	3.24	0.61	0.19	5.16	58	2.90	0.56	10.92	1.50	0.14	1.21
2	buputl	90	200	2.22	15	1.90	17	2.75	0.65	0.23	1.96	15	1.08	0.55	4.21	0.47	0.11	0.51
3	trace	53	66	1.62	6	2.23	14	2.62	0.24	0.09	7.42	68	5.12	0.69	10.62	1.30	0.12	1.17
4	dcg	34	287	8.44	63	15.47	93	3.58	1.26	0.35	3.33	14	0.67	0.20	11.21	0.45	0.04	0.00
5	botg	142	305	2.15	21	3.96	309	4.99	0.99	0.20	5.10	21	2.25	0.44	12.89	3.19	0.25	0.00
6	keitai	10	22	2.20	11	2.00	8	3.50	0.32	0.23	2.68	16	1.59	0.59	5.59	0.18	0.03	0.73
7	lfg	14	40	2.86	9	4.64	12	3.58	0.43	0.12	3.28	16	1.53	0.47	7.98	0.25	0.03	0.38
8	lfgutl	24	41	1.71	4	1.42	5	1.85	0.44	0.24	4.63	37	2.73	0.59	7.29	0.34	0.05	0.38
9	logtr	37	136	3.68	11	3.54	12	2.42	0.83	0.34	1.38	8	0.38	0.27	3.03	0.36	0.12	0.77
10	prover	39	81	2.08	4	2.21	8	2.74	0.54	0.20	2.54	17	1.35	0.53	4.95	0.41	0.08	0.46
11	rapilog	38	84	2.21	10	2.32	7	2.14	0.40	0.19	3.04	40	1.94	0.64	5.04	0.56	0.11	0.89
12	epilog	4	22	5.50	7	6.75	9	1.86	0.77	0.41	1.64	4	0.00	0.00	3.45	0.00	0.00	0.91
13	ppn	23	50	2.17	4	2.57	6	3.78	0.88	0.23	2.30	7	0.58	0.25	5.78	0.60	0.10	0.12
14	rubik	87	171	1.97	9	2.59	17	2.37	0.69	0.29	3.02	39	1.42	0.47	4.75	0.77	0.16	0.36
15	penta	27	43	1.59	4	2.00	11	2.12	0.86	0.41	2.30	19	0.79	0.34	5.19	0.70	0.13	0.26
16	assi	21	41	1.95	8	1.81	8	2.34	0.51	0.22	2.41	11	1.32	0.55	4.20	0.56	0.13	0.32
17	k-ac	17	39	2.29	8	1.71	7	2.03	0.44	0.22	4.08	18	2.97	0.73	3.64	0.59	0.16	0.33
18	sqlcm	38	103	2.71	10	2.11	8	2.82	0.85	0.30	2.78	12	1.02	0.37	6.42	0.65	0.10	0.45
19	rdb	31	88	2.84	15	3.23	24	3.72	0.97	0.26	2.49	10	0.98	0.39	6.31	0.68	0.11	0.59
20	typinf	20	49	2.45	11	2.00	7	2.06	0.41	0.20	2.96	17	1.80	0.63	4.53	0.63	0.14	0.63
21	esp1	62	173	2.79	10	2.58	12	3.10	0.49	0.16	1.90	8	0.94	0.49	4.27	0.47	0.11	0.71
22	esp2	76	154	2.03	6	2.64	17	3.34	0.73	0.22	2.47	19	0.82	0.33	6.05	0.75	0.12	1.00
23	esp3	37	78	2.11	11	1.41	6	2.27	0.33	0.15	2.53	24	1.28	0.51	4.32	0.97	0.23	0.96
24	asm	49	132	2.75	10	2.00	17	2.35	0.38	0.16	3.47	30	1.73	0.50	5.96	0.58	0.10	0.94
25	sub	49	121	2.47	6	2.04	9	2.96	0.49	0.16	2.07	19	1.17	0.56	4.64	0.57	0.12	0.83
26	def	36	498	13.83	155	8.53	149	3.86	0.59	0.15	0.89	9	0.15	0.16	3.26	0.06	0.02	1.00
27	psi	51	86	1.69	4	1.55	4	2.09	0.21	0.10	3.17	15	2.12	0.67	5.02	0.45	0.09	0.56
28	ope	42	128	3.05	10	2.90	19	3.30	0.52	0.16	2.02	11	0.92	0.46	4.61	0.27	0.06	0.98
29	plasy	44	90	2.05	4	2.27	5	2.77	0.89	0.32	1.80	12	0.32	0.18	4.37	0.13	0.03	0.72
30	analy	41	71	1.73	4	3.80	21	3.42	0.45	0.13	10.70	121	8.46	0.79	16.69	1.49	0.09	0.87
31	tree	27	68	2.52	5	2.74	25	3.90	0.72	0.18	2.97	11	1.84	0.62	6.68	1.16	0.17	0.78
32	cp-anal	60	129	2.15	24	1.95	8	1.98	0.60	0.30	4.73	46	2.90	0.61	7.62	1.05	0.14	0.84
33	cp	10	29	2.90	6	2.30	6	3.66	0.69	0.19	2.10	7	0.79	0.38	4.62	0.41	0.09	0.76

プログラム全体（今，プログラム個数は，33）についての収集dataの平均値の定義と値を次に示す。又，その幾つかに関しては，各プログラムの収集data値.vs.プログラム分布のグラフを付録に載せる。（Σは，プログラム全体についてのsummation ）

①平均OR relations

Σ Total OR relations ／Σ inference clause の種類数 =2.7

各プログラムの平均OR relation に注目したプログラム分布のグラフを付録 2-1に示す。

②平均参照数

Σ参照数／Σinference clauseの種類数 =3

③平均head predicate引数個数

Σ head predicate の引数個数／Σ Total OR relations =3.2

各プログラムのhead predicate引数の平均個数に注目したプログラム分布のグラフを付録 2-2に示す。

④head predicateの平均構造data引数ratio

Σhead predicateの構造データ引数個数／Σ head predicate の引数個数 = 0.2

head predicateの構造data引数ratio に注目したプログラム分布のグラフを付録 2-3に示す。

⑤平均AND literal 数

Σ AND literal数／Σ Total OR relations =3

1 inference clause当たりの平均AND literal 数に注目したプログラム分布のグラフを付録 2-4に示す。

⑥平均組込み述語数

Σ Evaluable prediates／Σ Total OR relations =1.4

⑦平均Evaluable predicate ratio

Σ ANDリララル数／Σ Evaluable predicate数 =0.5

Evaluable predicate ratio に注目したプログラム分布のグラフを付録 2-5に示す。

⑧平均body側引数個数

Σ body 側引数個数／Σ Total OR relations =6.5

各プログラムのbody側の引数の平均個数に注目したプログラム分布のグラフを付録 2-6に示す。

⑨body側の平均構造data引数個数

Σ Body 側構造data引数個数／Σ Body 側引数個数 =0.12

body側構造data引数ratio に注目したプログラム分布のグラフを付録 2-7に示す。

⑩平均CUT ratio

Σ CUT数／Σ Total OR relations =0.65

各プログラムのCUT ratio に注目したプログラム分布のグラフを付録 2-8に示す。

(2) 特性

プログラム全体の平均cut ratio は， 0.65 であり，付録 2-8より，大部分のプログラムは，deterministic に書かれていることがうかがえる。幾つかのプログラムにおいて，

cut ratio が 1 以上になるのは，

p(_ ,_):- ……，！；……，！；……！，

の形のclauseが要因となっている。プログラム全体のavORは， 2.7であり，平均cut ratio 0.65と合せ，解析したプログラムの多くは，実行時にAND-OR探索木の幅が，急激に

は，広がっていかないことが予想される。（なぜならば，OR relationsは複数あるが，成功するのは，多くの場合一つと考えられるから）もちろん，dcg program （プログラムNo４）のように高いOR relation 数を持つものもある。

avOR 2.7は，inference clauseの集合から，match する可能性のあるclauseを逐次的にpick up し，それを次々にunification するというpipeline的な処理をしても，ここ(match&unify) はbottle neck になりにくいことを示唆している。

付録２－１において，平均 OR relations が13～14のプログラムでは，

```
p(_, _) .
    :
    :            154個の並び
    :
p(_, _) .
p(_, _):- !, otherwise(_)
```

というようなclause群があり，これは，定義によりinference clauseとなる．
このことが，平均 OR relations を13～14にも押し上げている理由である．

平均参照数は，約 3であるが，最大参照数が10以上の解析対象プログラムが17と全プログラムの約半分在り，このことは，同一計算の共有（過去の参照結果を記憶しておき，同一の参照は，重複計算を回避して，過去の計算結果を利用する）メカニズム[19]の検討の余地を示唆している。

headの平均引数個数は，約 3なので，引数間のunification を並列化しても大きな効果は望めないであろう。

body側のひとつのAND literal 当たりの平均引数個数（Σbody側引数個数／ΣAND literals，Σはプログラム全体についてのsummation)は，約 2であり，これは，head predicate 当たりの平均引数個数の約 3に比べて小さい。これは，body側AND literalの約半分が組込み述語であり，組込み述語の引数個数の少なさに影響された結果である。

head predicate引数の内の約20% が構造体dataであり，又，body側引数の内約10% が構造体dataである。これは，静的な解析結果であり，静的にはvariableでも，実行時には，list data がinstantiate されることは往々にしてあり，動的には，この割合は増加するものと思われる。

3.2 inference clause 内組込み述語の解析結果と特性

(1) 解析結果

DEC-10 Prolog には広範な組込み述語が用意されており、付録３に，表１の33個のプログラムを解析して得られた組込み述語の使用頻度を示す。

(2) 特性

組込み述語はその実行機能により次のように分類される。

[01] Input/Output

(1) Reading in Programs
(2) File Handling
(3) Input and Output of Terms
(4) Character Input / Output

[02] Arithmetic
[03] Comparison of Terms
[04] Convenience
[05] Extra Control
[06] Information of the Program
[07] Meta-Logcal
[08] Modification of the Program
[09] Internal Databace
[10] Sets
[11] Compiled Program
[12] Debugging
[13] Definite Clause Grammars
[14] Environmental

この分類に従って組込み述語をリストアップしたのが付録４である。

なお，この分類は文献[18]に従った。

また，付録４中で/*U*/ 印のある組込み述語は，Unification の機能を使用するもの，もしくは使用した方がその実現が容易なものを表わす（他にもそのような組込み述語があるかもしれない）。

付録３，４から，組込み述語全体に対する主な実行機能別組込み述語の割合を計算したものが次の結果である。

[1] Input / Output	27.6%(1781)
[2] Meta-Logical	15.1%(977)
[3] Arithmetic	9.4%(608)
[4] Comparison of Terms	5.8%(372)
[5] Modification of the Program	4.7%(303)
[6] “＝”（使用頻度第１位）	14.1%(909)
[7] “mode”（使用頻度第２位）	11.1%(719)
	(Total:6455)
/*U*/ 印の組込み述語の全体に対する割合は	35.2%(2269)

また，“mode”と“public”を除いた上記の項目の割合は次の通りである。

[1] Input / Output	32.4%(1781)
[2] Meta-Logical	17.7%(977)
[3] Arithmetic	11.0%(608)
[4] Comparison of Terms	6.8%(372)
[5] Modification of the Program	5.5%(303)
[6] “＝”（使用頻度第１位）	16.5%(909)
	(Total:5505)

全AND literal の約50% が組込み述語だから，I/O 関連の組込み述語が全AND literal に占める割合は，約16% となり，DEC-10 Prolog プログラム実行を高速化するには，I/O を高速化する必要がある。

また，／*U* ／の付いた組込み述語の割合は全組込み述語の約41% となるから，Unification を必要としない組込み述語（除I/O ）の全AND literal に占める割合は約13% となる。よって，DEC-10 Prolog プログラム実行を高速化するには，I/O や unification を高速化するだけでなく，I/O 以外のunification を必要としない処理の負荷を算定し，この処理が全体に占める負荷が軽くないならば，これをも高速化する必要がある。

3.3 database clause解析結果と特性

(1) 解析結果

database clause に関する解析結果を表２に示す。

縦項目

表１の33個のプログラムに，database clause のみからなる以下の 6つのプログラムを挿入した。

dcg-db ：dcg プログラムに対する辞書databaseである。

botg-db ：bodgプログラムに対する辞書，non-terminalの停止条件等のdatabase

keita-d ：keitaiプログラムの辞書database

edb ：rdb プログラムの，fact名のdatabase

idb ： 〃 rule名 〃

anal-db ：組込み述語database

横項目

infe ：inference clauseの種類数

i-OR ：inference clauseのTotal OR relations

db ：database clause の種類数

d-OR ：database clause のTotal OR relation

T-kind ：infe+db

T-cla ： i-OR + d-OR

av-d-OR ： database clauseの平均OR relations

m-d-OR ：database clause の最大OR relations

av-d-H ：database clause の平均head predicate引数個数

d-H-SD/H：database clause の H-SD/H

(2) 特性

新たに入れた 6つのdatabase clause のみからなるプログラムのavORは10.0で最大ORは655とinference clauseのavOR 2.7， 最大OR 155に比べて約四倍程大きく，database clauseがさらに大規模化すれば，この倍率はもっと大きくなるであろう。よって，大規模database clause に関するunification の高速化がプログラム実行の高速化に大きな影響を及ぼす。（この６つのプログラムを含むdatabase clause を持つ29個のプログラムのavORは 8.7である。） 一方，これら６つのプログラムの平均head predicate引数個数は，2.8となり（29個のavHargは 2.6），inference clauseのそれとあまり変りない。

表２より，解析対象プログラムの中で，最大 inference clause 種類数は 142，また，最大 Total OR relations of inference clausesは 498，最大 database clause種類数は153，最大 Total OR relations of database clauses は1023である．各プログラムは実際にはリンクして使われるが，この数字はPrologマシンの実験機を作成する時の１つの指

計となるであろう.

3.4 静的解析時の制限

(1) このstatic analizer では，引数の内部構造は原則として解析されない。
例えば次のような四つのclauseからなるプログラムがあったとする。

```
bc(X,Y):-b_cw(X,Y).
lte(X,Y):-apply([t!,bc,tr,ld,tr,lu,tl],X,Y). ── ①
apply([],X,X).
apply([Op! Opseq],X,Y):-z=..[Op,X,X1],call(Z),!,apply(Opseq,X1,Y).
```

このプログラムを静的に解析した結果，predicate bc（_ ,_）の参照数は 0になる。なぜならば，apply の定義の中まで追っていかないと，clause①内のbcの引数が２だということは分らない。本static analizerでは，head predicate symbol と引数個数が一致したとき，その述語が参照されていると判定するので，（head predicate symbol が等しく，引数個数の異なるpredicate は一つのプログラム内でともに存在することはありうる。）①からだけでは，bc(X,Y) が参照されているとは判定できない。よって静的な解析では，bc(X,Y) が参照されていることを検出することは難しい。

表2．database clause 解析結果

No.	Name	infe	i-OR	db	d-OR	T-kind	T-cla	av-d-OR	m-d-OR	av-d-H	d-H-SD/H
1	buptr	50	90	0	0	50	90	0.00	0	0.00	0.00
2	buputl	90	200	3	4	93	204	1.33	2	2.50	0.00
3	trace	53	86	3	3	56	89	1.00	1	1.00	0.00
4	dcg	34	287	0	0	34	287	0.00	0	0.00	0.00
5	dcg-db	0	0	13	109	13	109	8.38	21	4.00	0.50
6	botg	142	305	0	0	142	305	0.00	0	0.00	0.00
7	botg-db	0	0	153	1023	153	1023	6.69	655	2.66	0.16
8	keitai	10	22	1	1	11	23	1.00	1	1.00	0.00
9	keita-d	0	0	3	422	3	422	140.67	253	3.46	0.30
10	ifg	14	40	4	4	18	44	1.00	1	4.50	0.22
11	ifgutl	24	41	5	5	29	46	1.00	1	1.20	0.00
12	logtr	37	136	3	12	40	148	4.00	6	1.33	0.25
13	prover	39	81	1	2	40	83	2.00	2	2.00	0.50
14	rapiog	38	34	0	0	38	84	0.00	0	0.00	0.00
15	epilog	4	22	6	24	10	46	4.00	4	3.17	0.43
16	ppn	23	50	0	0	23	50	0.00	0	0.00	0.00
17	rubik	87	171	22	67	109	238	3.05	35	2.49	0.60
18	pento	27	43	6	114	33	157	19.00	63	2.47	0.63
19	assi	21	41	6	17	27	58	2.83	6	1.24	0.76
20	k-ac	17	39	4	17	21	56	4.25	8	1.88	0.78
21	salcm	38	103	4	12	42	115	3.00	6	3.83	0.26
22	rdb	31	88	4	27	35	115	6.75	14	3.04	0.20
23	edb	0	0	4	24	4	24	6.00	7	2.00	0.00
24	idb	0	0	1	22	1	22	22.00	22	1.00	1.00
25	typinf	20	49	0	0	20	49	0.00	0	0.00	0.00
26	esp1	62	173	4	32	66	205	8.00	18	2.00	0.02
27	esp2	76	154	2	7	78	161	3.50	4	3.00	0.43
28	esp3	37	78	0	0	37	78	0.00	0	0.00	0.00
29	asm	48	132	0	0	48	132	0.00	0	0.00	0.00
30	sub	49	121	3	96	52	217	32.00	62	2.00	0.00
31	def	36	498	5	5	41	503	1.00	1	1.00	0.40
32	psi	51	86	2	78	53	164	39.00	62	2.00	0.00
33	ope	42	128	3	17	45	145	5.67	9	1.53	0.38
34	plasy	44	90	0	0	44	90	0.00	0	0.00	0.00
35	analy	42	72	0	0	42	72	0.00	0	0.00	0.00
36	anal-db	0	0	1	153	1	153	153.00	153	1.00	0.78
37	tree	27	68	2	44	29	112	22.00	43	1.16	0.84
38	cp-anal	60	129	2	2	62	131	1.00	1	1.00	1.00
39	cp	10	29	1	2	11	31	2.00	2	3.00	0.17

4. おわりに

DEC-10 Prolog プログラムを静的に解析し各種dataを収集した結果，

①DEC-10 Prolog は逐次実行型 Prolog であり，メモリ空間が十分でない。programmerは，そのことを意識してプログラムを記述している。そのためDEC-10 Prolog プログラムは deterministicになる傾向（高いcut ratio ）がある。（しかしこのことは解析対象プログラムが扱っている問題に並列性がないこと，あるいは，Prolog自体に並列性がないことを示している訳ではないことを一言つけ加えておく。）

②AND literal の約半分は組込み述語であり，その組込み述語のうち約30% はI/O である。よって，I/O の速度がプログラム実行速度に影響を及ぼす。

③database clause のOR relation 数は inference clause のOR relation 数(約３）の約４倍であり，database clause が大規模化するとこの比は増大する。よって大規模database clause を含むプログラムの場合，database clause に関するunificationの高速化がPrologプログラム実行の高速化に大きな影響を及ぼす。

等が，考察された。これらは，DEC-10 Prolog マシンを設計する際の一助となる。

最後に，日頃ご指導ご鞭撻をいただく村上国男第一研究室長，解析対象プログラムを提供し，議論して下さったICOT研究員の方々，およびE.Y.Shapiro 氏(Weizmann Institute of Science, Israel)に深謝する。

－参考文献－

[1] 平川秀樹，三吉秀夫，他，“論理プログラミングによるボトムアップパーサ(BUP)トランスレータの開発”，情報処理第２６回全国大会．

[2] 三吉秀夫，他，“Prologによる日本語文節DCG の生成”，情報処理第２７回全国大会．

[3] 松本裕治，他，“Prologに埋め込まれたボトムアップパーサ：BUP ”，Logic Programming Conference '83,Tokyo.

[4] 安川秀樹，“LFG in Prolog －Toward a formal system for representing grammatical relations －”，ICOT Technical Report TR-019 (1983.8).

[5] 向井国昭，他，“Prologによる線形順序戦略定理証明プログラム”，情報処理第２７回全国大会．

[6] 坂井公，宮地泰造，“Prologへの否定的知識の導入とプログラム検証”，ICOT Technical Report TR-007 (1983.5).

[7] 古川康一，“Prologによる問題解決”，情報処理第２３回全国大会．

[8] 古川康一，近藤浩康，“Two Dimensional Programming in Prolog ”，Logic Programming Conference '83, Tokyo.

[9] 國藤進，他，“Prologによる対象知識とメタ知識の融合とその応用”，ICOT Technical Report TR-009 (1983.5) .

[10] 宮地泰造，他，“論理データベース向きの知識同化方式の一提案”，ICOT Technical Memo TM-0004 (1983.4).

[11] 横田治夫，他，“RDBM Delta（Ⅲ）＝コマンド体系について＝”，情報処理第２６回全国大会．

[12] 横田治夫，他，“An Ennanced Inference Mechanism for Generating Relational Algebra Queries ”，ICOT Technical Report TR-026 (1983.10) .

[13] 古川康一，竹内彰一，“論理型言語における type inference とその type check への応用”，情報処理第２６回全国大会．

[14] 近山隆．　，“パーソナル逐次型推論マシンψ－その言語システム－”，情報処理第２７回全国大会．

[15] T.Chikayama, “ESP －Extended Self-contained PROLOG－ as a Preliminarg Kernel Language of Fifth Generation Computers ”，New Generating Computing,1 (1983).

[16] 高木茂行，“汎用型マイクロプログラム・アセンブラ”，ICOT Technical Report TR-021 (1983.8)．

[17] E.Y.Shapiro ，“A subset of Concurrent Prolog and its Interpreter ”，ICOT Technical Report TR-003 (1983.1)

[18] D.L.Bowen, “DEC system-10 Prolog User's Manual”, Dept of Artificial Intelligence, University of Edinburgh (1981).

[19] 平川秀樹，尾内理紀夫，古川康一，“POPS:OR-Parallel Optimizing Prolog System”，情報処理第２７回全国大会．

付録1．解析対象プログラムの機能概要一覧

番号	プログラム名	概　要
1 2	BUP トランスレータ BUP トランスレータの ユーティリティ [1],[3]	BUP トランスレータは，文法規則の入った DCGファイルを入力とし，Prologプログラムのファイルを生成する．この際に次に示す処理を行う． (1) 文法規則をBUP 形式のProlog節に変換する． (2) 全ての文法カテゴリに対するターミネイト節を生成する． (3) 文法カテゴリ間のLINK関係を計算してリンク節を生成する．
3	BUP トレーサ [1],[3]	BUP プログラムのトレーサであり，次の機能を有する． (1) ステップバイステップにルールを適用して行く． (2) パージングの状況に関する情報を表示する． (3) 規則の提供に関するコントロール・コマンドを受けつける． プログラムはPrologインタプリタ部，情報表示部及びパージングステップの記憶部より構成される．
4	文法記述(DCG) [2]	DCG(Definite Clause Grammar) の記法で書かれた文法規則でBUP トランスレータの入力となる．
5	文法記述(BUP) [1],[3]	BUP トランスレータにより生成されたBUP 形式の文法規則で，Prologの実行形式ファイルである．
6	形態素解析プログラム [2],[3]	簡単な日本語文節の形態素解析（語尾処理，自動分かち書き）プログラムの半自動生成及び形態素解析の実行を行うプログラムである．

番号	プログラム名	概　要
7 8	LFG システム LFG システムの ユーティリティ [4]	BUP システムを用いて自然言語文の構文解析を行なうとともに，文の機能構造を抽出するプログラムである． LFG:Lexical Functional Grammar
9	述語論理式簡単化 プログラム	与えられた一階述語論理式をスコーレム化しさらに積和標準形に変換するプログラムである．出力の標準形は定理証明プログラムの入力となる．変換のついでに論理法則に従った最適化を行っている． $a \vee a = a$ $a \wedge (b \vee a) = a$ 等々である． アルゴリズムはいわゆるdivide and conquer式に行なっている．たとえば$a \wedge b$に対する処理はaとbをそれぞれ独立に行なった結果をマージする方式である． スコーレム関数名の生成(gensym)にassert/retractを使用している．
10	線形順序戦略定理証明 プログラム [5]	一階述語論理の完全な定理証明プログラムである．証明方法はOrdered Linear法と呼ばれるものである．証明の探索法は下降型混合探索である． 一度現われた中心節は覚えておき同じ計算は避けている．そのためにassert/retractを使用している． プログラムの主要部はユニフィケーション処理，レゾリューション処理である．

番号	プログラム名	概　要
11	無限木Prolog インタプリタ	infinite tree も扱うPrologインタプリタである．サイクルを含んだ有向グラフをデータとして直接扱うPrologプログラミングを楽しむことができる． 例題として不完全に与えられた条件から，その条件を満たす最小状態数を持つ有限オートマトンの合成問題等がある． プログラムはユニファイヤとインタプリタとから成る．infinit treeのunification のために内部的に特別なタームの形式を用いている．
12	Epilogインタプリタ	ロジックプログラミングにおけるメタ制御を記述することができるインタプリタである．coroutine 制御，lazy評価，シーケンシャル制御が記述できる．
13	depth-first PPN インタプリタ [6]	Prologに普通の意味での否定（cut を使って定義するnegation as failure ではない）を導入した言語(PPN)の逐次型（ depth-first）インタプリタ． PPN:Pure Prolog with Negation

番号	プログラム名	概　要
14	ルービック・キューブ [7]	simple production systemを用いたルービック・キューブの解法を求めるプログラム．決定的な処理を行う．プロダクション・ルールは35個．
15	ペントミノプログラム [8]	「ペントミノ」と呼ばれるゲームの解をすべて求めるプログラム．
16	知識同化プログラム [9]	例外値を含む知識同化プログラムをメタ推論用demo述語を用いて実現したもの．
17	知識獲得プログラム [10]	知識獲得方式のデモプログラムである．知識獲得方式としては演繹論理志向の知識の同化という概念を，オブジェクト言語とメタ言語の融合という考え方に基いて実現可能な形で定式化した．知識同化の概念は，証明可能性，矛盾性，冗長性，および独立性というサブ概念のチェックと対応する内部データベース更新とからなる．また，論理データベース向きの知識同化のインプリメントを，論理型プログラム言語PROLOGにより行った．その結果，PROLOGが知識の同化の実現などに非常に適していることがわかった．
18	SEQUEL- 関係代数 インタプリタ [11]	関係データベース問合せ言語SEQUELの問合せを関係代数ベースのコマンド列に落とし，関係代数コマンド実行シミュレータにより求める関係を得る．

番号	プログラム名	概　要
19	Prolog－関係データベース・インタフェースプログラム [12]	Prologプログラムのfactの大部分を関係データベースに格納するとした時のPrologと関係データベースのインタフェースを取るプログラムである．1つのPrologのゴール文を，内部のデータベースのみで推論すると同時に，外部の関係データベースにあるfactに対する問合せプランを生成し，そのプランを関係代数ベースのコマンド列に落とし，シミュレータでそのコマンドの実行をシミュレートするものである．
20	タイプ推論システム [13]	Prologプログラムをデータとして受取り，そのプログラム中の各述語の各引数のタイプ（データ構造）を推論するプログラムである．
21	ESP クロスシステムのユーティリティ・プログラム [14],[15]	ESP クロスシステムのための種々のutility ． 以下のものを含む． (1) Macro Expander (2) Pretty Printer ('write'+αのことをする) (3) formatted output (writef) (4) 種々のutility （append等）
22	ESP クロスシステムのプリプロセッサ・プログラム [14],[15]	ESP クロスシステムのPreprocessor部． ESP プログラムを読んでDec-10 Prolog のプログラムに翻訳するpre-compiler．

番号	プログラム名	概　要
23	ESP クロスシステムのランタイム・サポートプログラム [14],[15]	ESP クロスシステムのRuntime Support 部. Object-oriented calling mechanism の実現や，debugging utility , PSI の機械語KL0 をDEC10 上に実現するためのcode等を含む.
24	汎用型アセンブラのメインプログラム [16]	汎用型アセンブラの本体部分. アセンブラ・ソースのプログラムの読込み，及び制御を行なう.
25	汎用型アセンブラのサブルーチンプログラム [16]	汎用型アセンブラのサブルーチン部分. 10進→16進変換 ファイル名の読込み（プロンプト）等.
26	汎用型アセンブラの定義プログラム [16]	汎用型アセンブラの機械依存定義部分. ニーモニックの定義 フィールドの定義 コード表の定義etc.
27	分岐アドレス編集プログラム	PSI 用μアセンブラの生成する分岐方法を入力し，PSIハードウェアの内蔵しているμ命令の多方向分岐制御用メモリの内容を生成するプログラム.
28	演算子順位構文解析プログラム	演算子順位に従って構文解析を行ない，構文木（項といってもよい）を出力するプログラム．現在のレベルは，Prologの構文解析プログラムとほぼ同等.

付録A　組込み述語の分類

[01] Input / Output

(1) Reading-in Programs

```
consult(_)                  reconsult(_)            [_|_]
```

(2) File Handling

```
see(_)              /*U*/ seeing(_)              seen
tell(_)             /*U*/ telling(_)             told
close(_)                 fileerrors             nofileerrors
rename(_,_)              log                    nolog
end_of_file
```

(3) Input and Output of Terms

```
/*U*/ read(_)                write(_)            display(_)
     writeq(_)          /*U*/ print(_)
```

(4) Character Input / Output

```
nl                          get0(_)             get(_)
skip(_)                     put(_)              tab(_)
ttynl                       ttyflush            ttyget0(_)
ttyget(_)                   ttyskip(_)          ttyput(_)
ttytab(_)
```

[02] Arithmetic

```
(_+_)                       (_-_)               (_*_)
(_/_)                       (_ mod _)           -(_)
(_/\_)                      (_\/_)              \(_)
(_<<_)                      (_>>_)              !(_)
$(_)                /*U*/ (_ is _)             (_=:=_)
(_=\=_)                     (_<_)               (_>_)
(_=<_)                      (_>=_)
```

[03] Comparison of Terms

```
(_==_)                      (_\==_)             (_@<_)
(_@>_)                      (_@=<_)             (_@>=_)
compare(_,_,_)              sort(_,_)           keysort(_,_)
```

[04] Convenience

```
(_,_)                       (_;_)                   true
fail                /*U*/ (_=_)             /*U*/ length(_,_)
```

[05] Extra Control

```
repeat              /*U*/ \+(_)              /*U*/ (_->_)
```

[06] Information about the state of the Program

```
     listing        /*U*/ listing(_)         /*U*/ numbervars(_,_,_)
/*U*/ ancestors(_)  /*U*/ subgoal_of(_)      /*U*/ current_atom(_)
/*U*/ current_functor(_,_)                   /*U*/ current_predicate(_,_)
```

[07] Meta-Logical

```
       var(_)                    nonvar(_)                 atom(_)
       integer(_)                atomic(_)         /*U*/ functor(_,_,_)
 /*U*/ arg(_,_,_)          /*U*/ (_=..._)                name(_,_)
 /*U*/ call(_)
```

[08] Modification of the Program

```
       assert(_)                 asserta(_)                assertz(_)
 /*U*/ clause(_,_)          /*U*/ retract(_)         /*U*/ abolish(_,_)
```

[09] Internal Database

```
 /*U*/ recorded(_,_,_)            recorda(_,_,_)            recordz(_,_,_)
 /*U*/ erase(_)             /*U*/ instance(_,_)            assert(_,_)
       asserta(_,_)                assertz(_,_)       /*U*/ clause(_,_,_)
```

[10] Sets

```
 /*U*/ setof(_,_,_)         /*U*/ bagof(_,_,_)             (_^_)
```

[11] Compiled Program

```
       compile(_)          /*U*/ incore(_)          /*U*/ revive(_,_)
      (mode _)                   (public _)                fastcode
       compactcode
```

[12] Debugging

```
 /*U*/ unknown(_,_)               debug                     nodebug
       trace                     leash(_)                  spy _
       nospy _                   debugging
```

[13] Definite Clause Grammars

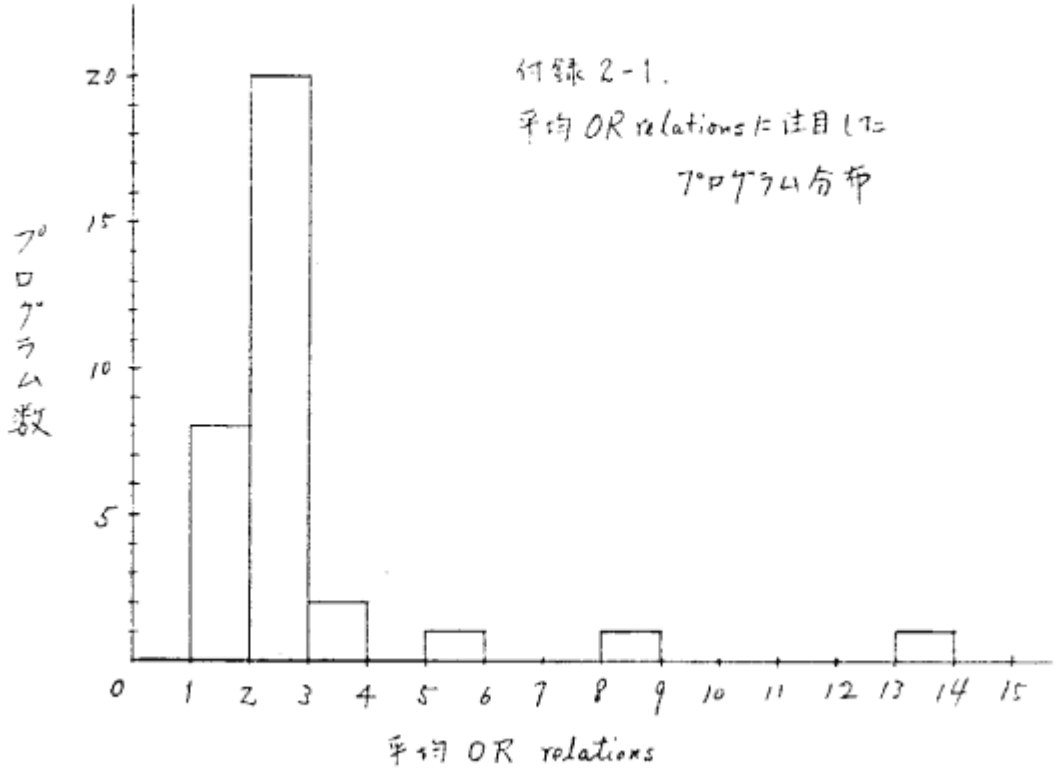
```
       (_-->_)                   expand_term(_,_)          phrase(_,_)
```

[14] Environmental

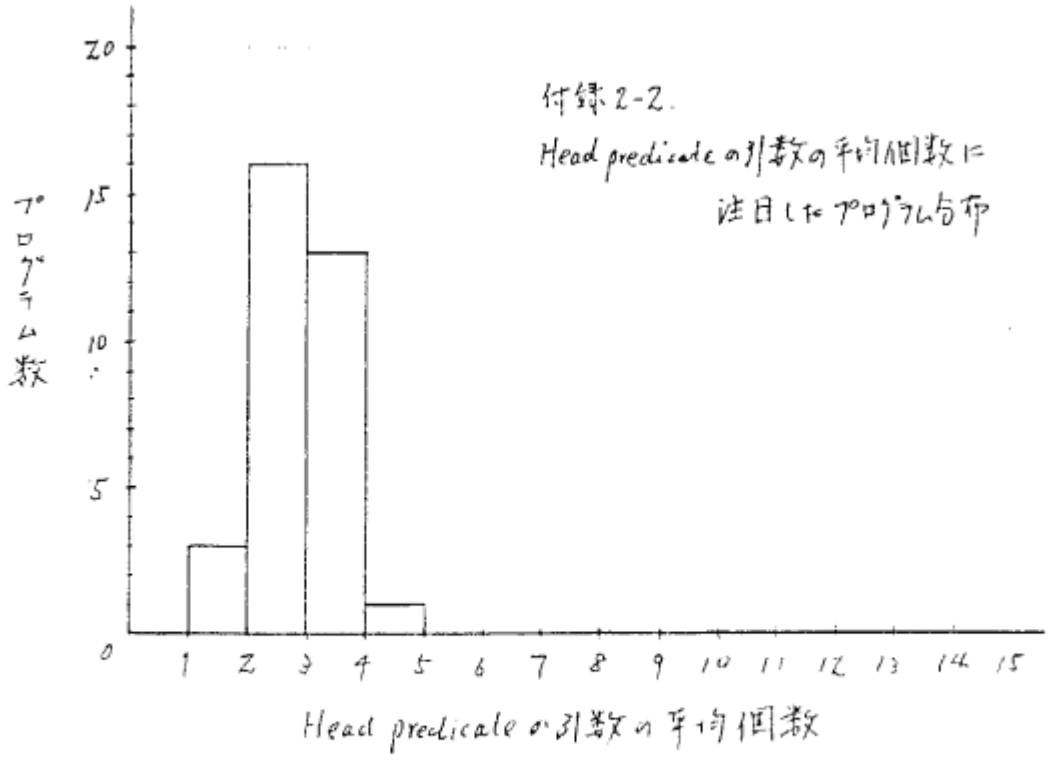
```
       version                   version(_)                reinitialise

       'NOLC'                    'LC'                      cp(_,_,_)
  /*U*/ current_op(_,_,_)  /*U*/ prompt(_,_)

       gc                        nogc              /*U*/ gcguide(_,_,_)
       trimcore

       break                     abort                     halt

       statistics          /*U*/ statistics(_,_)           maxdepth(_)
 /*U*/ depth(_)

       save(_)                   save(_,_)                 restore(_)

       plsys(_)                  core_image                run(_,_)
       tmpcor(_,_,_)
```

番号	プログラム名	概　要
29	PLA 圧縮プログラム	1ビットデコード方式のPLA の簡単化を行うプログラム．deterministic なプログラムであり，list構造を頻繁に操作している．
30	DEC-10 Prolog プログラム・スタティック・アナライザ	DEC-10 Prolog プログラムを静的に解析し、各clauseのOR関係数，clauseの参照数，ヘッド述語内引数個数，ボディ内引数個数，AND リテラル数，組込み述語数，カット数等を調べ，各平均を求め，出力する．
31	AND-OR探索木特性アナライザ	解析対象プログラムとこのプログラムに対する質問を入力として，その場合の探索木（AND-OR Tree)をAND は逐次に，ORは並列に解析し，各レベルにおける木のnode数，branch数等を出力する． （簡単なOR並列シミュレータとみなすこともできる）．
32	Concurrent-Prolog アナライザ	Concurrent-Prolog プログラムと，goalを入力とし，sub-goalの状態（suspend，dead，reducible，active etc.），および状態に関する統計情報を出力するアナライザ．
33	Concurrent-Prolog インタプリタ [17]	Concurrent-Prolog のインタプリタである．

付録2.



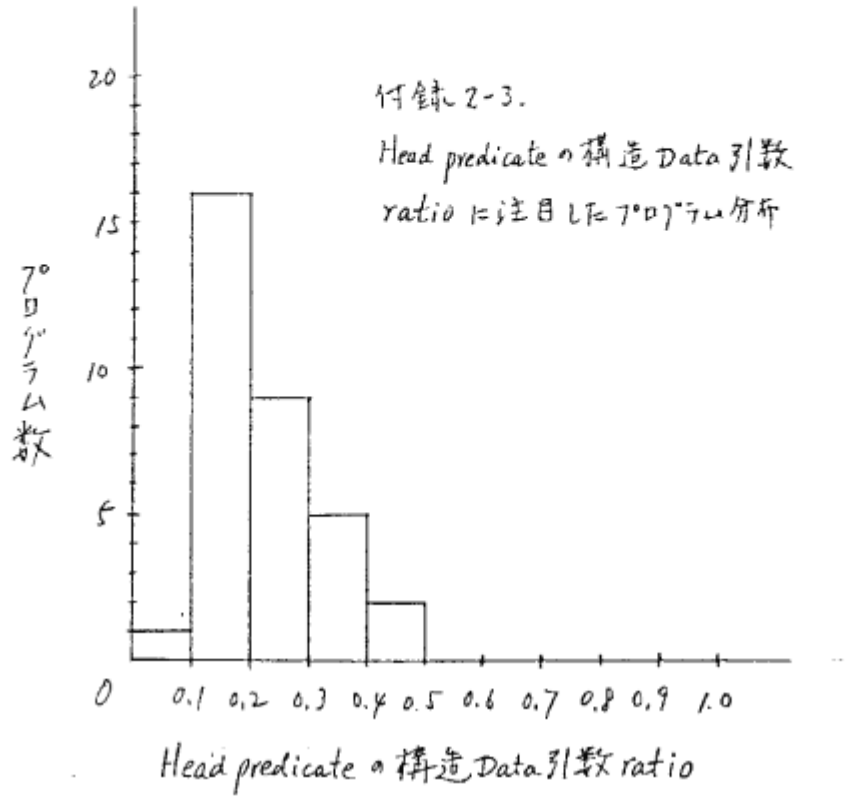
付録 2-1.
平均 OR relations に注目した
プログラム分布



付録 2-2.
Head predicate の引数の平均個数に
注目したプログラム分布

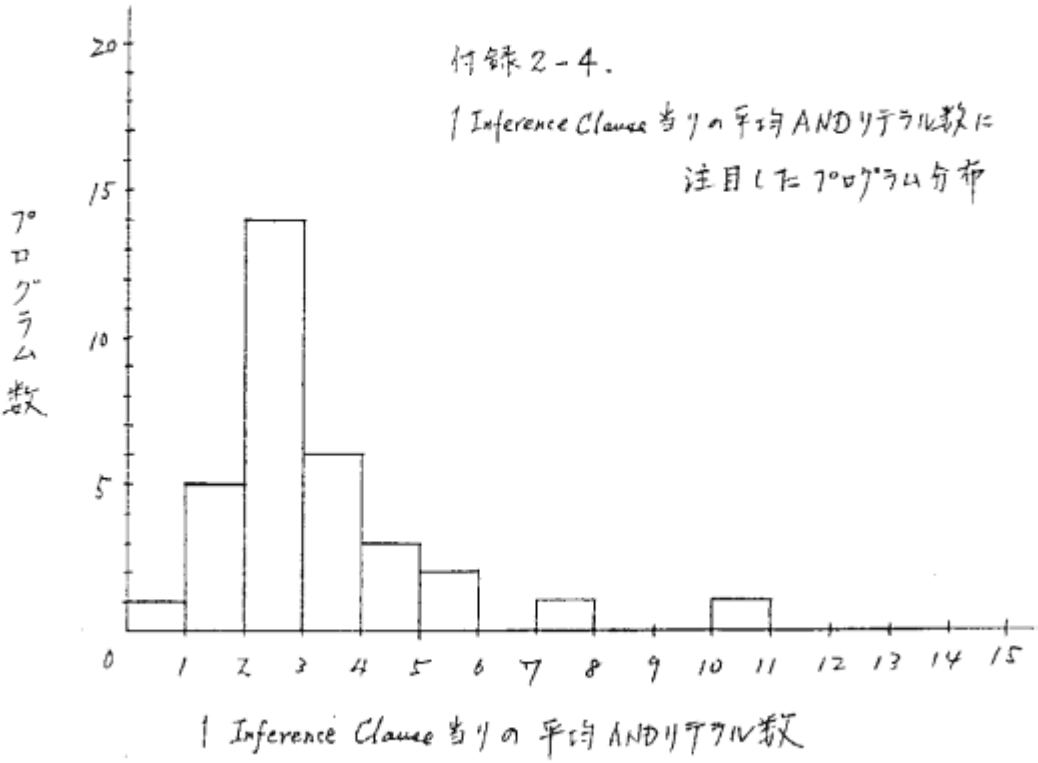
付録2-3.

Head predicateの構造Data引数
ratioに注目したプログラム分布

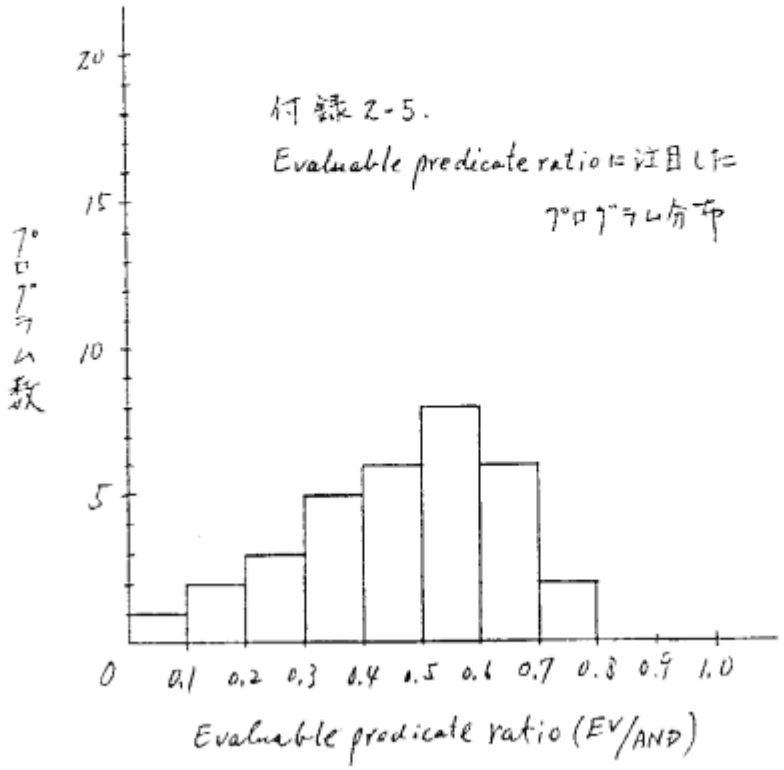


付録2-4.

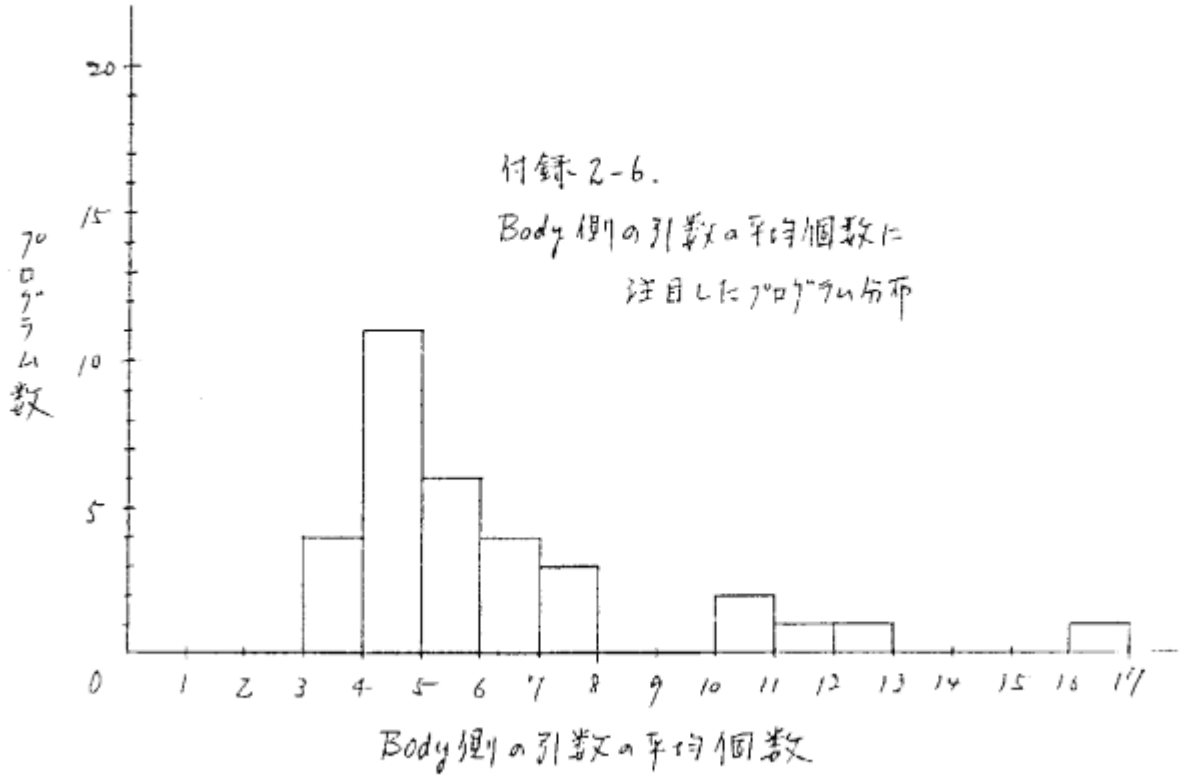
1 Inference Clause当りの平均ANDリテラル数に
注目したプログラム分布



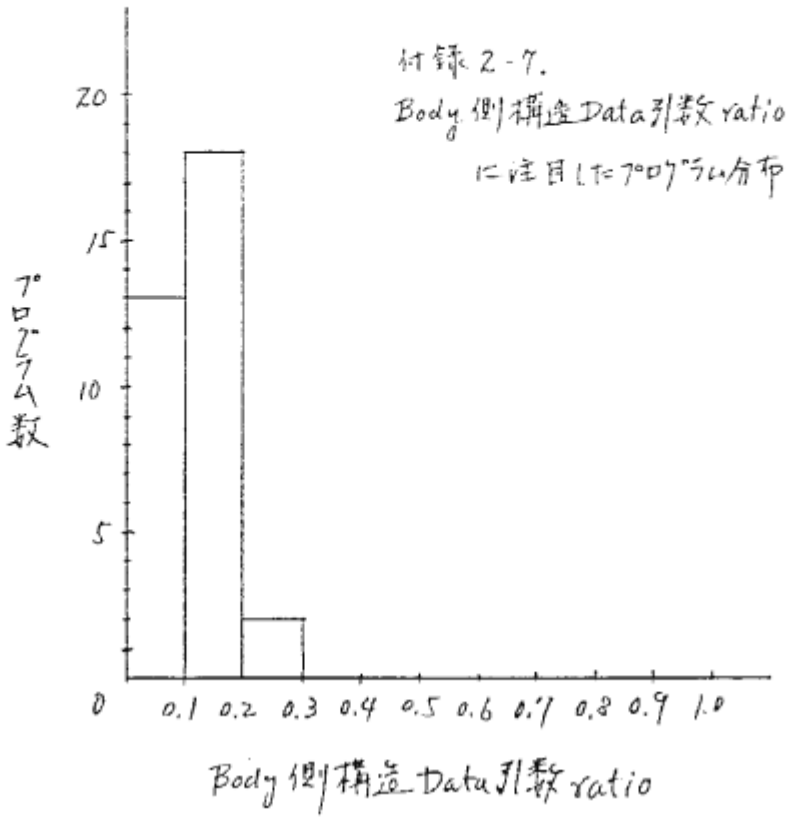
付録 2-5.
Evaluable predicate ratio に注目した
プログラム分布



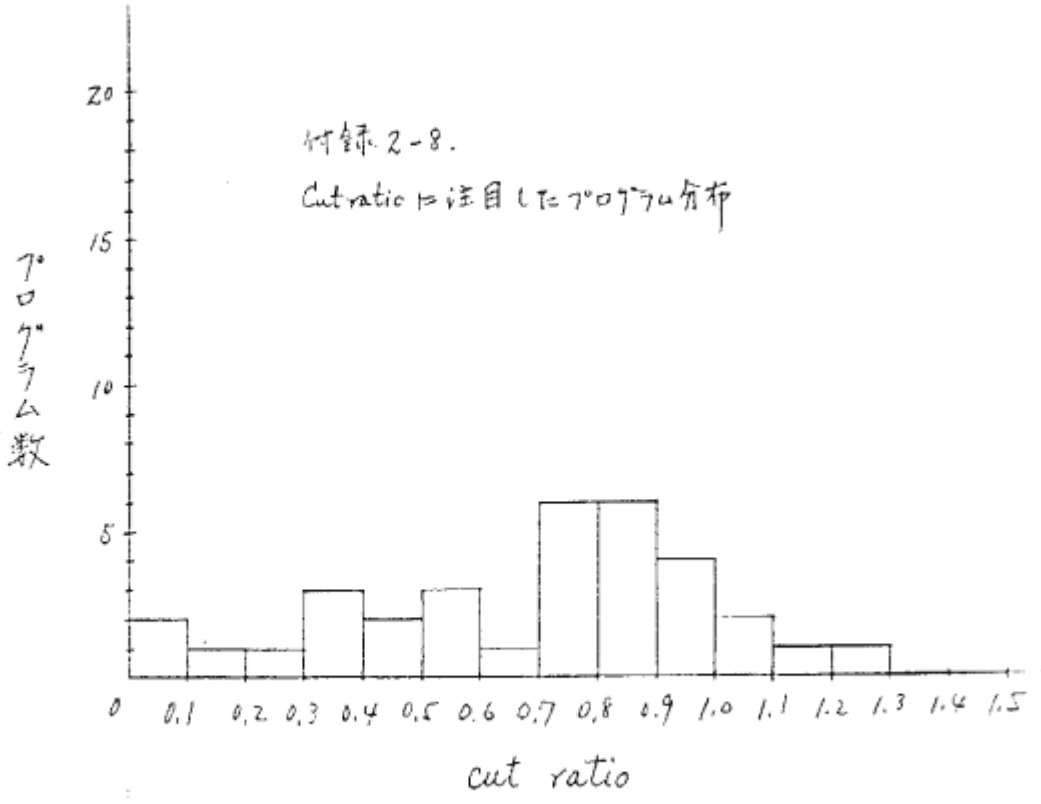
付録 2-6.
Body 側の引数の平均個数に
注目したプログラム分布



付録 2-7.
Body 側構造 Data 引数 ratio
に注目したプログラム分布



付録 2-8.
Cut ratio に注目したプログラム分布



付録３．組込み述語の使用頻度

No.	Entered	total
1	=	909
2	mode	719
3	nl	487
4	display	428
5	call	354
6	==	343
7	write	293
8	is	244
9	public	230
10	=..	222
11	+	154
12	fail	136
13	op	132
14	tab	122
15	ttyni	102
16	var	97
17	arg :3	85
18	abolish	82
19	asserta :1	79
20	- :2	78
21	functor	71
22	name	67
23	true	64
24	put	62
25	retract	59
26	assert :1	59
27	ttyflush	56
28	tell	52
29	atomic	38
30	>	35
31	read	32
32	<	30
33	length	29
34	\+	27
35	print	26
36	current_op	24
37	statistics :2	23
38	get0	22
39	clause :2	19
40	see	17
41	nonvar	16
42	=<	15
43	>=	15
44	seen	14
45	integer	14
46	telling	13
47	atom	13
48	/	12
49	\==	12
50	told	10
51	close	10
52	repeat	10
53	recordea	9
54	*	8
55	@=<	8
56	promot	8
57	ttyget0	8
58	numbervars	8
59	mod	8
60	->	7
61	abort	7
62	trimcore	7
63	get	6
64	halt	6
65	assertz :1	5
66	recorda	5
67	listing :1	5
68	erase	4
69	compile	4
70	fileerrors	4
71	save :1	4
72	restore	4
73	break	4
74	rename	4
75	.[_|_]	3
76	sort	3
77	fastcode	3
78	writeq	3
79	reconsult	3
80	<<	3
81	bagof	3
82	trace	3
83	gc	3
84	- :1	2
85	@<	2
86	incore	2
87	nodebug	2
88	seeing	2
89	@>	2
90	=\=	2
91	nofileerrors	2
92	LC	1
93	compactcode	1
94	NOLC	1
95	spy	1
96	@>=	1
97	setof	1
98	statistics :0	1
99	\/	1
100	skip	1
101	recordz	1
102	current_functor	1
103	nolog	1
104	keysort	1
105	=:=	1
106	ttyget	1
107	plsys	1