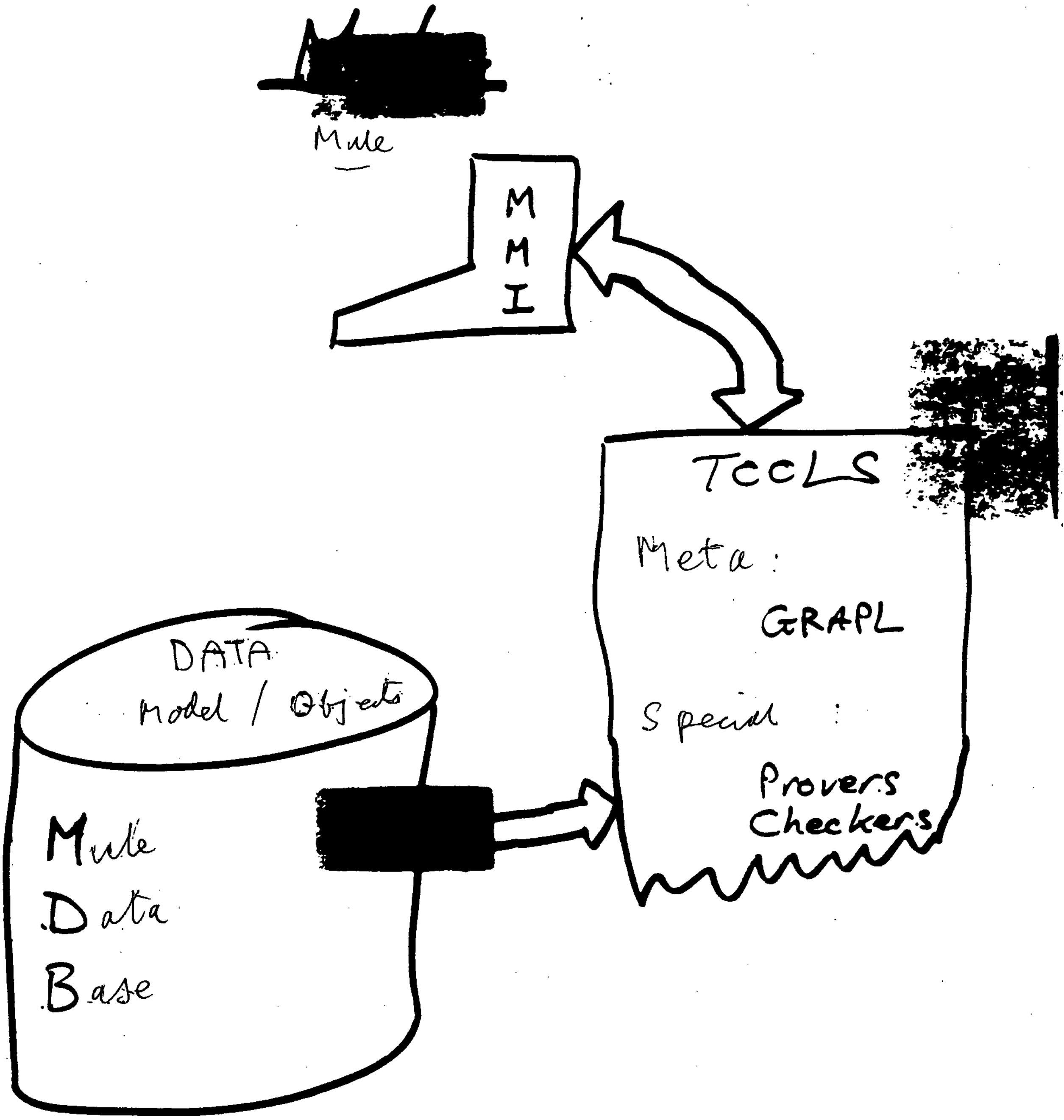
BCS [redacted] - FACS

85.12.17

Ian Cottam

Lecture 2

Grap[redacted]l + Rapid Prototyping


[redacted]
Mule
MMI
TOOLS
Meta :
GRAPL
Special :
Provers
Checkers
DATA
Model / Objects
Mule
Data
Base

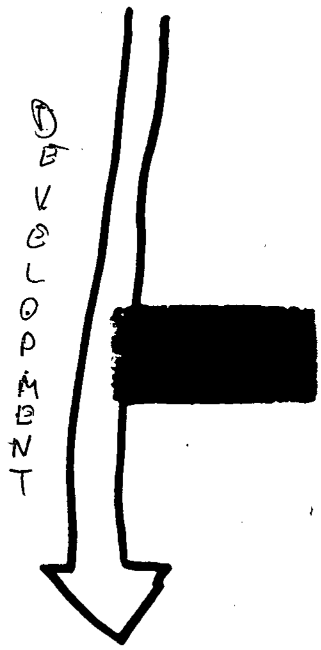
Data [redacted] objects – MDB

Model

BRM

General [redacted]

query oriented



[redacted]

Specific Application constrained queries

Directed Graphs ⊕ ...

MDB supports:

- Labelled, [redacted], graphs
  - nodes/arcs labelled by entities
  - nodes may contain values
  - one-one (nodes, entities)
  - many-one (nodes, values)
  - many-one (arcs, entities)

'Node' and '[redacted]' are synonymous

+

- Lists
- Sets
- Value maps

} convenience

• DATA MANIPULATION LANGUAGE

[redacted]

"GRAPL"

Experimental [redacted]

VHLL for database (MDB)

Query and Update [redacted]

- Prolog
- Typol (Mentor)

- support [redacted] activities beyond the context-free
  - type checking
  - formula transformat[redacted]
  - proof checkers

[redacted]

Q. Why Prolog-like?

[redacted]

need ...

- logical inferences on MDB
- (subgraph) pattern-match ing [redacted]
  & symbolic manipulations
- ability to handle incomplete data (e.g. syntax graph [redacted] "holes")
  with

[+ not much time to design/ "re-invent"]

Grapl [redacted] Prolog

- almost identical ("to our surprise [redacted]
- all Grapl predicates are typed.
- concep[redacted]. type inheritance.
- several trivial details.

Sub-type [redacted] e.g.

```
person :: A:Age, N:Name;
student :: person + C:Dept,
                    U:Univ[redacted]

young(person(?AGE,_))
      <= ?AGE < 21.

?- young(stud[redacted]3,_,_,_)).
```

yes

ABSTRACT SYNTAX

Formula [redacted]

Formula :: TERM : Term;

Term :: EXPR : Expr;

Expr = vari | true | false | undef [redacted]
or | neg | impl | turnsti[redacted]

vari :: VAR : Id;

true :: ; false :: ; undef :: ;

or :: O1 : Exp[redacted] O2 : Expr;

and :: A1 : [redacted] A2 : Expr;
Expr

impl :: A : Expr, C : Expr;

neg :: NOT : Expr;

turnstile :: LHS : Expr, RHS : Expr;

[redacted]

```
(* Grapl prog. A.S. *)
check :: T: Term;
copy :: INE: Expr, INB: List,
        OUTE: Expr, OUTE: List;
taut :: E: Expr, L: List;
falsify :: E: Expr, L*: List;
eval :: E[redacted], V: Expr;
isin :: I: Id, V: Expr, L: List;
writeVal :: L: List;

triple :: I: Id, V: Expr, L: List
List :: triple | nil;     nil :: ;
```

[redacted]

PropLogic 3:

check (Term (?E)) <=

copy(?E, nil, ?C[redacted]

taut (?C, ?B).

check

ATTACH [redacted] TO Term

WITH "Tautology check".

```
              ?B
taut (?E, [redacted] <=
    not ( falsify (?E, ?B) ),
    write (<'TAUTOLOGY!'>),
    writeln.

falsify (?E, ?B) <=
    eval (?E, false),
    write (<'False with valuation'>),
    write( [redacted]
    writeVa (?B).

(* ditto for eval (?E, undef) *)
```

false

eval(true[redacted]e). eval(false, false)
eval(undef, undef). eval(vari(?v), ?v)

eval(and(?E1, _), false) <=
    eval(?E1, false).
eval(and(_, ?E2), false[redacted]
    eval(?E2, false).
eval(and(?E1, _), undef) <=
    eval(?E1, undef).
eval(and[redacted]E2), undef) <=
    eval(?E2, undef).
eval(and(?E1, ?E2), true) <=
  eval(?E1, true), eval(?E2, true

(* ditto [redacted]

eval {
- or
- impl
- turnstile
- neg

[redacted]

```
copy (vari( ?[redacted] ?ENV, vari(?V), ?ENV)
  <= isin ( ?I, ?V, ?ENV), !.
copy (vari(?I), ?ENV, vari(?V),
          triple(?I, ?V, ?ENV)[redacted]
copy(true, ?I, true, ?I).
copy (false, ?I, false, ?I).
copy (undef, ?I, undef, ?I).
copy (and(?E1, ?E2), ?I, and(?C1, ?C2), ?O2)
   <= copy(?[redacted] ?C1, ?O1),
        copy (?E2, ?O1, ?C2, ?O2).
(* ditto for copy of or, impl,
   turnstile and neg *)
```

```
isin(?I, [redacted]iple(?I, ?V, _)).
isin(?I, ?V, triple(_, _, ?L)) <=
    isin(?I, ?V, ?L).

writeVal(triple(?I, ?V, ?L)[redacted]
    write(<?I, '=', ?V>),
    writeln,
    writeVal(?L).

writeVal([redacted]
```

Limitatio[redacted]f Current Ver

- cannot update database.
  - need assert/retrac[redacted]

- sub-typing ("inheritance") is too slow.
  - need[redacted]t implementation

    Smart

- all Grapl output goes to a monitor window (not current)
  - needs fixing.

Rapid Prototyping of V[redacted]M Specifications.

D

Why?

[redacted]

- early customer feedback:
  - would this functionality satisfy your requirements? [redacted]
- helps specifier/designer:
  - is spec. consistent,
    computable,
    [redacted] etc. ?

(primitive theorem proving assistance)

D[redacted]ER!

- encourages TEST-CASE approach to correctness

– management reluctant to have staff "wasting time" proving theorems [redacted] Spec./Prototype passes all Q.A's tests.

Credits :
IDC [redacted] White +
Shirley McAse
(

Goal

The automatic (or, at least, semi-automatic) generation of a prototype implementation from a [redacted] formal spec.

(N.B. not possible for all VDM specs.)

PROBL[redacted]

- past

  $\exists i \in \mathbb{N} \cdot$

  $\neg \exists j \in \mathbb{N} \cdot j > i$ [redacted]

  (* "there is a largest value in the set of naturals *)

- $\forall i \in \mathbb{N}$ [redacted] $even(i) \vee odd(i)$

  univ. quant. over infinite set.

[redacted]

Fundamental Requirement

A person writing an abstract formal spec. in VDM s[redacted] not have to know about or be concerned about the prototyping [redacted]ocess.

⟹ better for prototype generation to fail than spec. is "infected".

VDM [redacted] – spec.

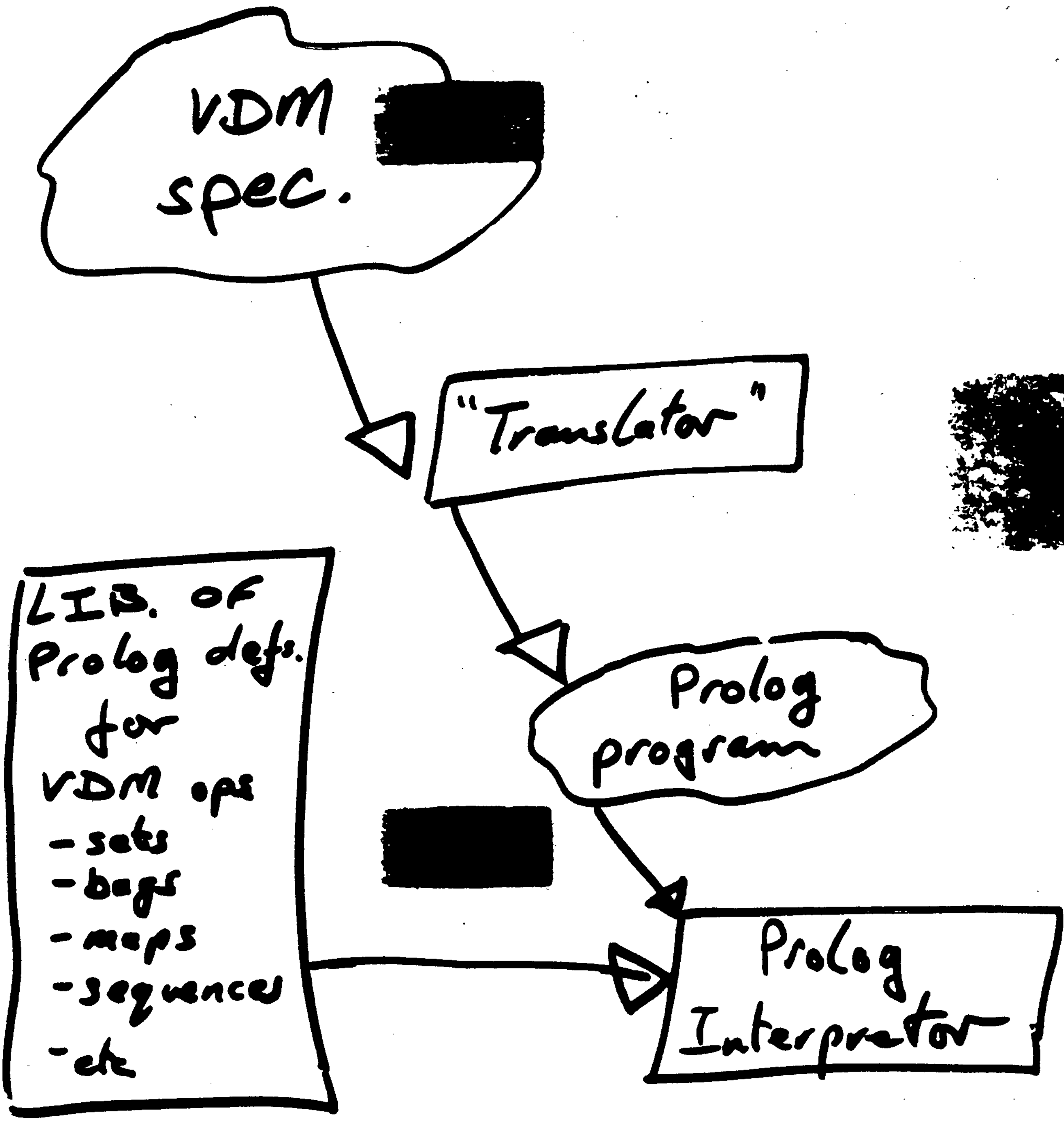
pred. logic
relational
(non-determinis

[redacted]

↓

Prolog [redacted]

pred. logic
(Horn cl
relational
(non-deterministi


VDM [redacted]
spec.
"Translator"
Prolog
program
LIB. of
Prolog defs.
for
VDM ops
- sets
- bags
- maps
- sequences
- etc
[redacted]
Prolog
Interpretor

e.g.

Implicit [redacted] [defn.] of seq. concat.

VDM: concat: seq of X × seq of X
→ seq [redacted] X

post-concat(l_1, l_2, r) [redacted]
len r = len l_1 + len l_2 [redacted]
r(1 ... len l_1) = l_1 ∧
r(len l_1 + 1 ... len r) = l_2

Prolog:

```
concat(L1, [redacted]) :-        [L2, R]
    len(L1, [redacted]), len(L2, LENL2),        [LENL1]
    LENR is LENL1 + LENL2, len(R, LEN
    subl(R, 1, LENL1, RF), RF = L1,
    MID = LENL1 + 1, subl(R, MID, LENR, RB),
    RB = L2.
```

VDM		Prolog
	[redacted]	
sets		lists (unique)
sequences		lists
maps		lists of 2-[redacted] (unique 1^{st} e[redacted]
bags		lists of 2-lists ("map X to $\mathbb{N}$")
	[redacted]	
State Variables		Prolog's facts/rules Data Base

VDM p[redacted] conditions

```
post_OP ( ... ) :-        [redacted]
        state(Old),
          ↑
        translated post-cond
          ↓  (post( Old, .., New
        r[redacted]ct ( state(_),
        asserta ( state(New),
```

e.g. The TEST [redacted] from the "Equiva[redacted] Relation" problem.

VDM: Part = *set of set of* Pno

TEST (p1, p2 : Pno) r : $\mathbb{B}$

ext rd p : Part

post $r \Leftrightarrow \exists s \in \overleftarrow{p} \bullet p1 \in s \wedge p2 \in s$

Prolog:

```
test(P1, P2) :-
    stat[redacted], member(S, P),
    member(P1, S), member(P2, S).
(* No state change *)
```

The EQUATE op.

[redacted]

VDM

EQUATE $(p1, p2 : Pno)$

<u>ext</u> <u>wr</u> $P : Part$

<u>post</u> $P =$

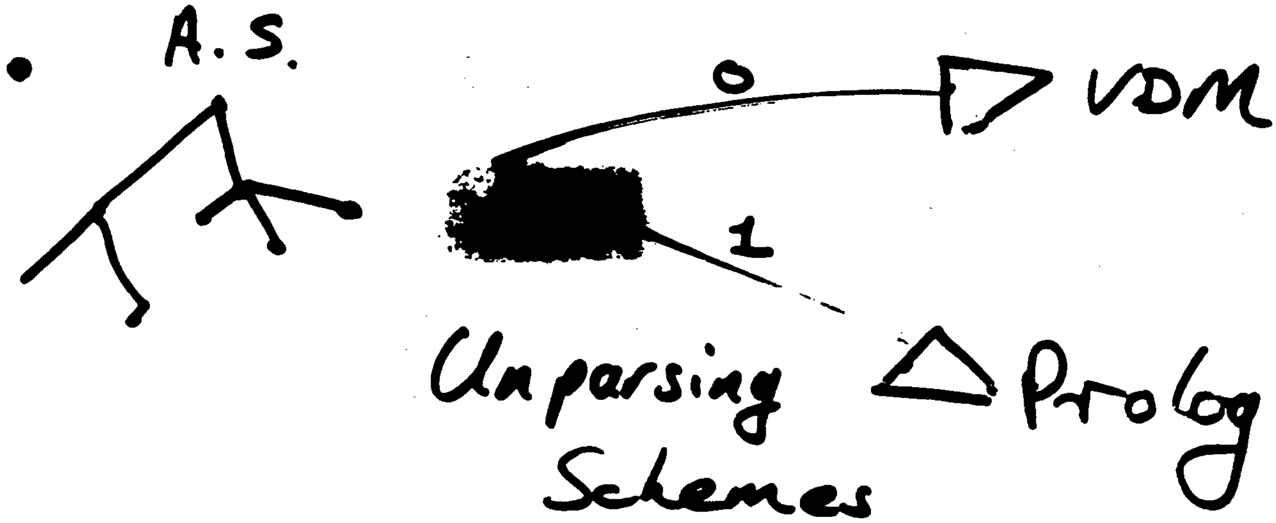
$$\{s \in \overleftarrow{P} \mid p1 \notin s \land p2 \notin s\}$$

$$\cup \; \{\bigcup \{s \in \overleftarrow{P} \mid p1 \in s \lor p2 \in s\}\}$$

[redacted]

Current Implementation

- Semi-automatic translation, based on a VDM-editor generated via the C.[redacted] C.M.U. ALOEGEN programs.

- 

Future Plans

- Rule-b[redacted] system
  - fully automatic
  - action routines [redacted]
    GrapL producing [redacted]

- General investigation of Rapid [redacted]totyping within IPSE 2.5 Alvey project.

Mule :

[redacted]

- single-user, workstation-based, "IPSE"

- ICL / Perq Systems PERQ I [redacted]

  - UNIX
  - Pascal (UNIX-Portable)

[redacted]

- VDM version

Major [redacted]ences:

- Gandalf CMU (USA)
- Mentor INRIA (Fr[redacted]
- PSG Darmstadt (Germany)

+ "Workstation [redacted]
concepts Xerox PARC (USA)

Related Work

[redacted]

Zs. Farkas, P. Szeredi, and E. Santane-Toth,

"LDM - a program specification [redacted] support system.",

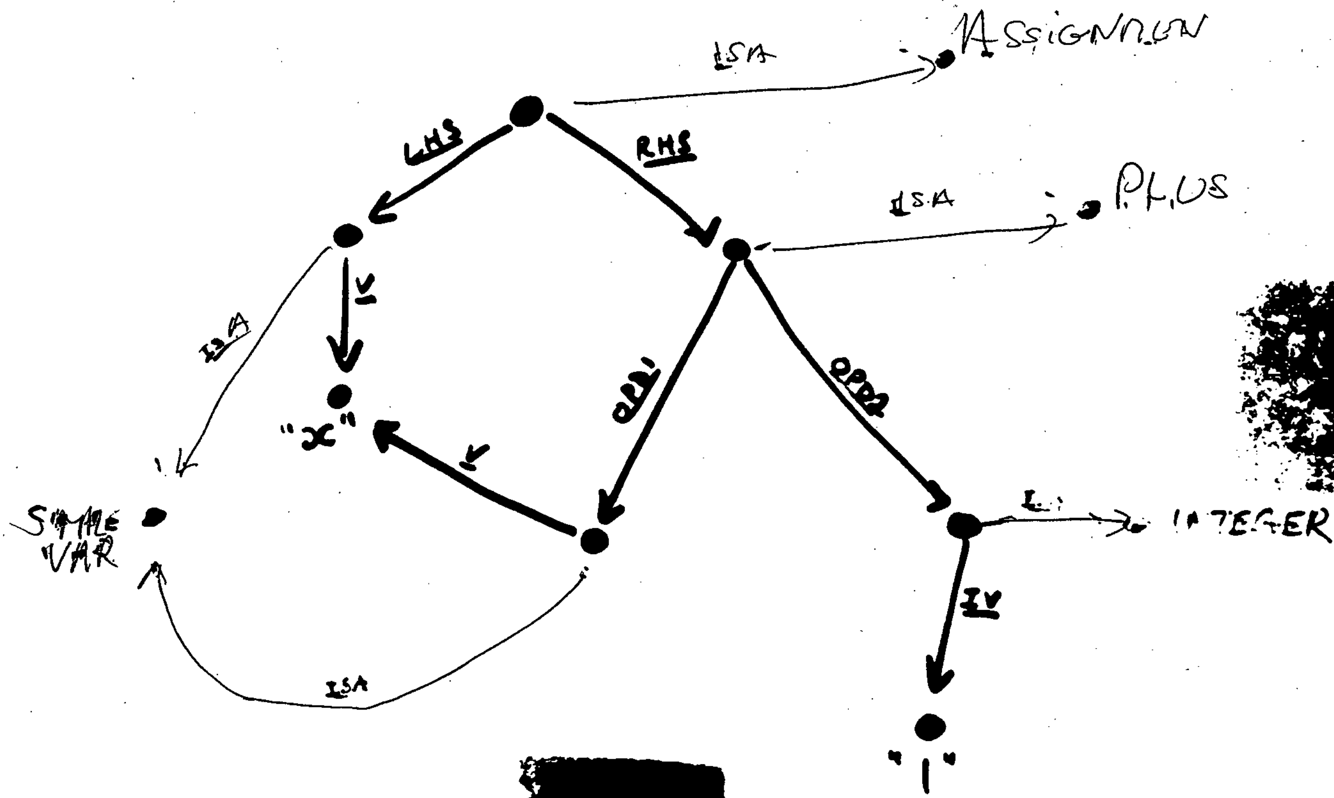
Proc. 1st Int. Conf. on Logic Prog.

Marseille, 1982.

[redacted]

x +

x := [redacted] 1



[redacted]

[redacted]

Q. How to make graph links?

- MMI pointing device
- "Dynamic syntax" for text I/P
- Explicit action routine in some DBL.

$E1$ —[redacted]→ $E3$

$E1 \xrightarrow{E2} S = \{F1, \ldots, Fn\}$ [redacted]

$E1 \xrightarrow{E2} L = \{F1, \ldots, Fn\}$

E_v —[redacted]→ $VS = \{F1, \ldots, Fn\}$