Revised Report on the Algorithmic Language ALGOL 68

A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck,
C.H.A. Koster, M. Sintzoff, C.H. Lindsey,
L.G.L.T. Meertens, R.G. Fisker.

A van. Wijngaarden ... two level grammars
M. Sintzoff ... nest syntax L. Meertens ... predicates

A language with two sentences

he sings a song

we sing a song

Meta rule

A) NUMBER :: singular ; plural.

Hyper-rules

a) NUMBER sentence:
NUMBER subject, NUMBER predicate.

b) NUMBER subject: NUMBER pronoun.

c) NUMBER predicate: NUMBER verb, object.

d) object: "a song".

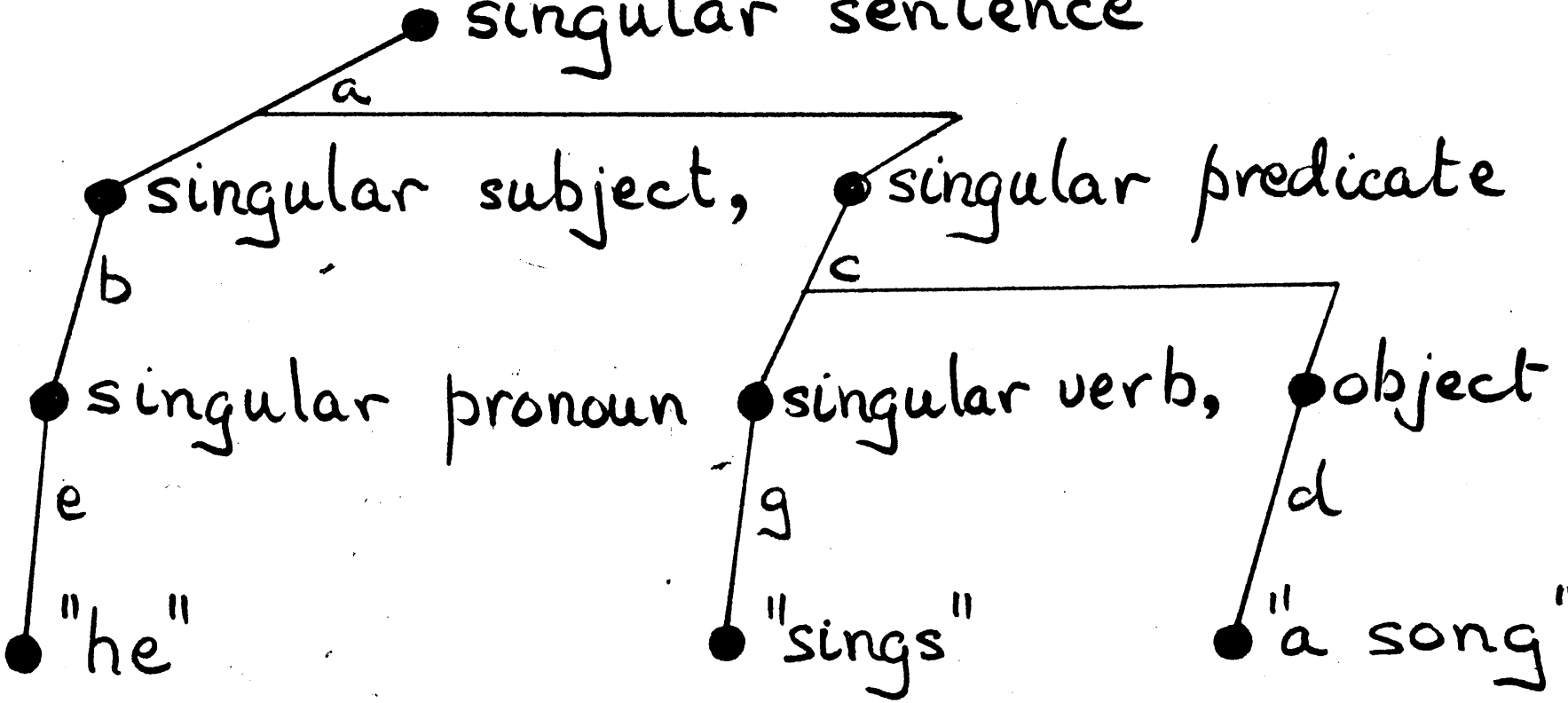
e) singular pronoun: "he".

f) plural pronoun: "we".

g) singular verb: "sings".

h) plural verb: "sing".

Parse tree for "he sings a song"



Metarules

A) PRONOUN :: he ; we.

B) VERB :: sing ; sings.

Hyper-rules

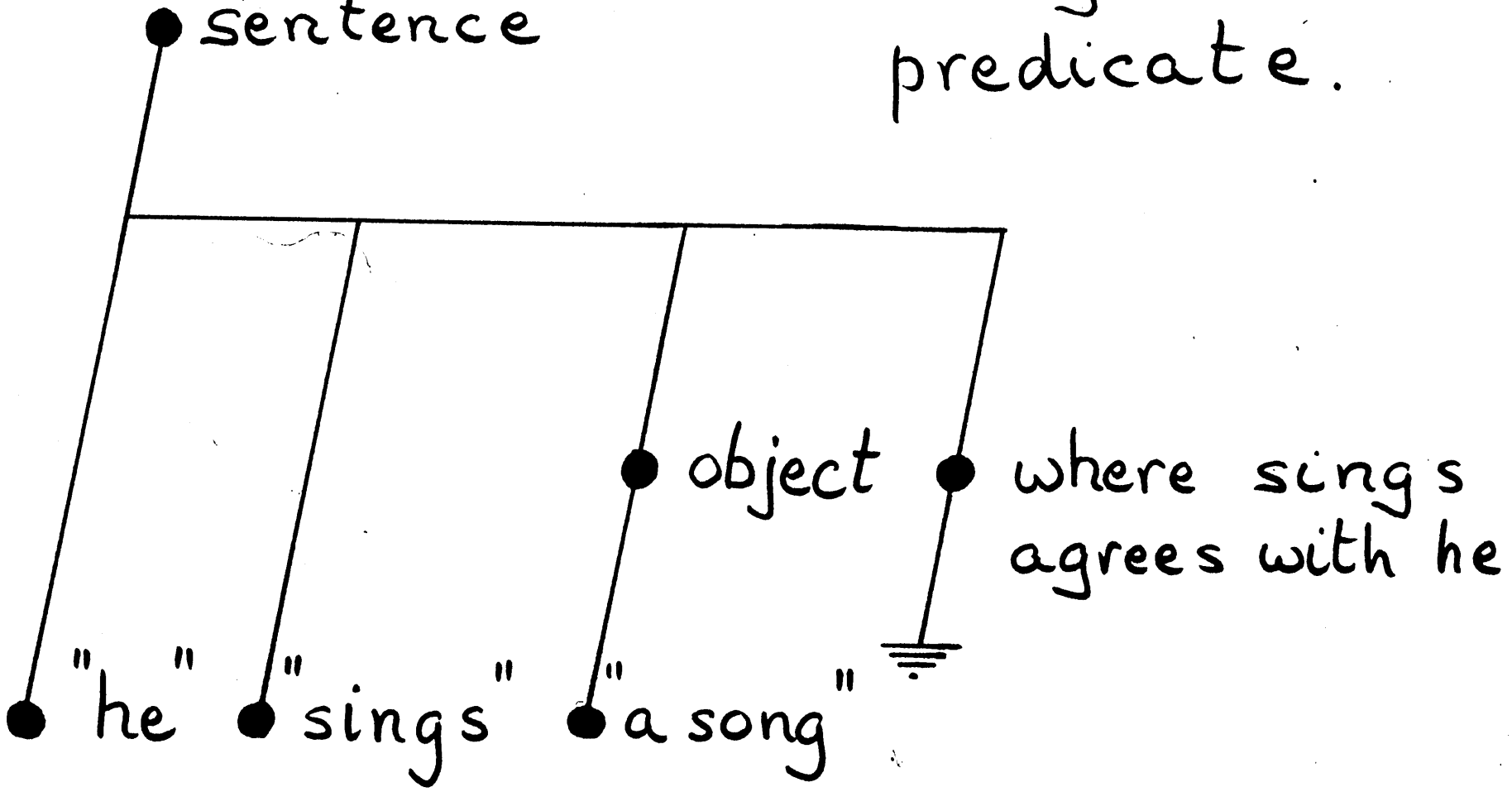
a) sentence : "PRONOUN", "VERB", object,
where VERB agrees with PRONOUN.

b) object : "a song".

c) where sings agrees with he :.

d) where sing agrees with we :.

Parse tree using a predicate.



ALGOL 68

General predicates

A) ALPHA:: a; b; c; d; e; f; g; h; i; j; k; l; m;
n; o; p; q; r; s; t; u; v; w; x; y; z.

B) NOTION :: ALPHA ; NOTION ALPHA.

C) NOTETY :: NOTION ; EMPTY.

D) EMPTY ::.

E) THING :: NOTION ; <NOTETY> NOTETY;
THING <NOTETY> NOTETY.

F) WHETHER :: where ; unless.

Predicate hyper-rules

a) where true :.

b) unless false :.

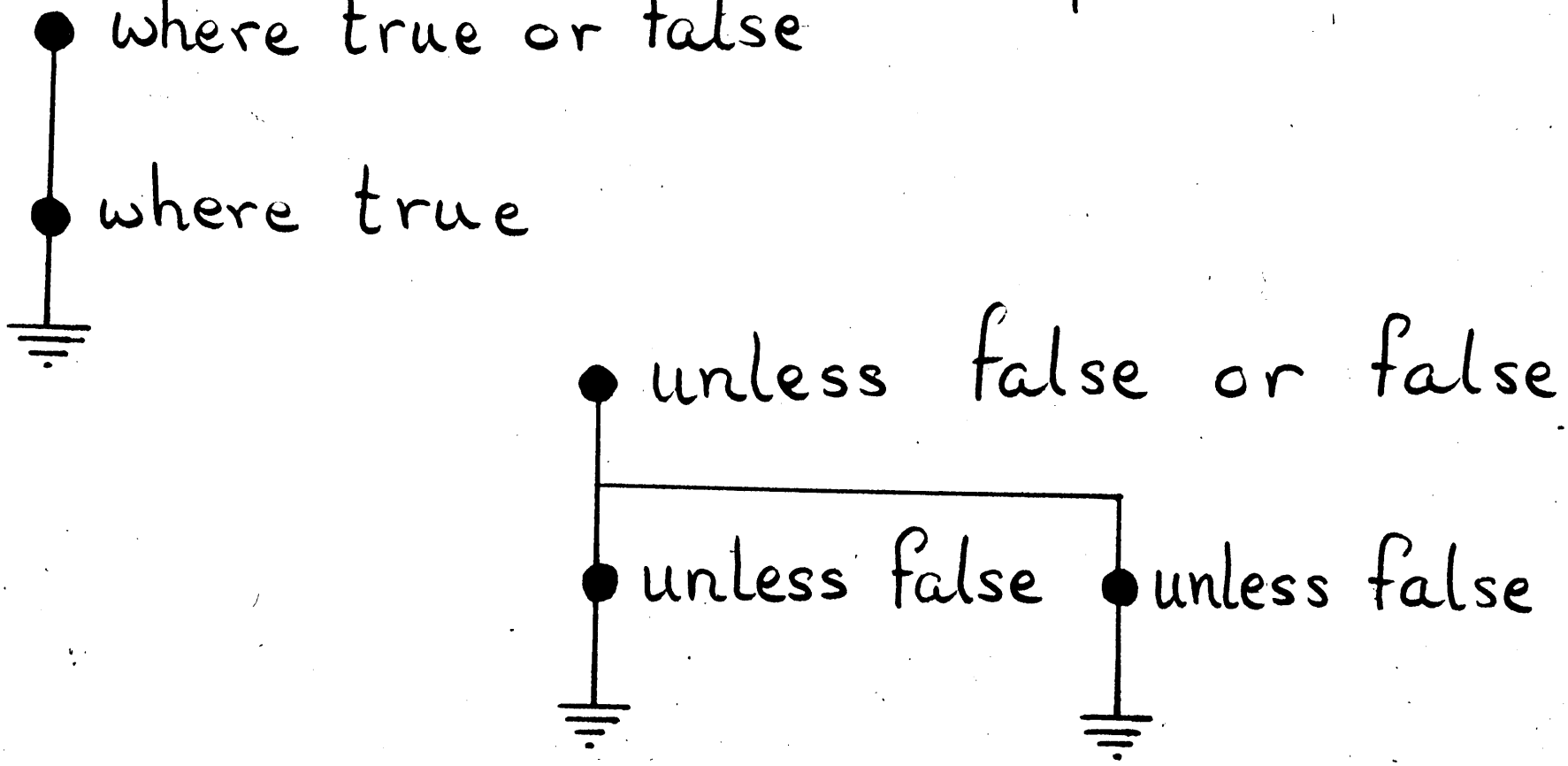
c) where THING1 and THING2 :
    where THING1, where THING2.

d) where THING1 or THING2 :
    where THING1 ; where THING2.

e) unless THING1 and THING2 :
    unless THING1 ; unless THING2.

f) unless THING1 or THING2 :
    unless THING1 , unless THING2.

Production trees for predicates



String matching

) WHETHER <NOTETY1> is <NOTETY2>:

WHETHER <NOTETY1> begins with <NOTETY2>

and <NOTETY2> begins with <NOTETY1>.

begins with

h) WHETHER <EMPTY> begins with <NOTION>:
    WHETHER false.

i) WHETHER <NOTETY> begins with <EMPTY>:
    WHETHER true.

j) WHETHER <ALPHA1 NOTETY1>
        begins with <ALPHA2 NOTETY2>:
    WHETHER <ALPHA1> coincides with <ALPHA2>
      in <abcdefghijklmnopqrstuvwxyz>
      and <NOTETY1> begins with <NOTETY2>.

coincides with

k) where <ALPHA> coincides with <ALPHA>
in <NOTION> : where true.

l) unless <ALPHA1> coincides with <ALPHA2>
in <NOTION> :

where <NOTION> contains
<ALPHA1 NOTETY ALPHA2>
or <NOTION> contains
<ALPHA2 NOTETY ALPHA1>.

contains

m) WHETHER <ALPHA NOTETY>
contains <NOTION>:
WHETHER <ALPHA NOTETY>
begins with <NOTION>
or <NOTETY> contains <NOTION>.

n) WHETHER <EMPTY> contains <NOTION>:
WHETHER false.

ALGOL68 Mode declarations

```
mode lr = long real ;

mode vector = [1:n] real ;

mode a = union (int, real, char);

mode cell =
   struct (real val, int n, ref cell next)
```

Ill-formed modes

```
mode a = a,
```

```
mode a = b, b = a,
```

```
mode d =
struct (int n, d filler)
```

TALLY :: i ; TALLY i.

actual <TALLY1> declarer :

where <TALLY1> is <i>, actual declarator;

where <TALLY1> is <TALLY2 i>,

TALLY2 mode indication.

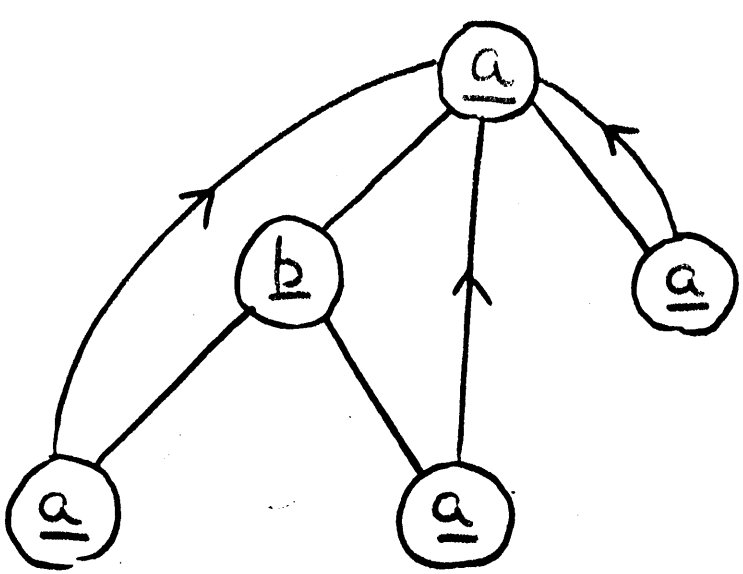
mode a = real, b = a, c = b

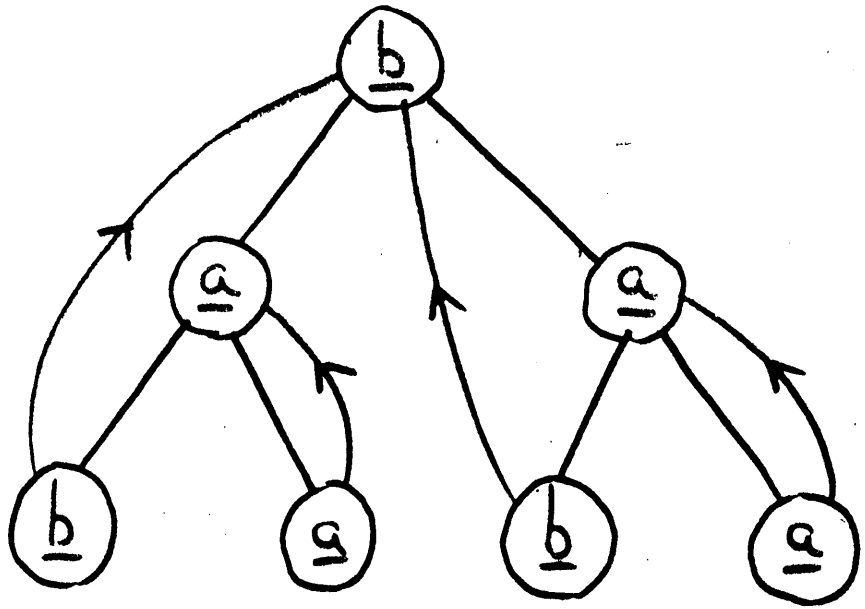
TALLY 1 2 3 ... actual-declarer

TALLY 1 2 3 .... mode-indication

Recursive Modes

```
mode a = struct (ref b x, ref a y),
       b = struct (ref a x, ref a y)
```





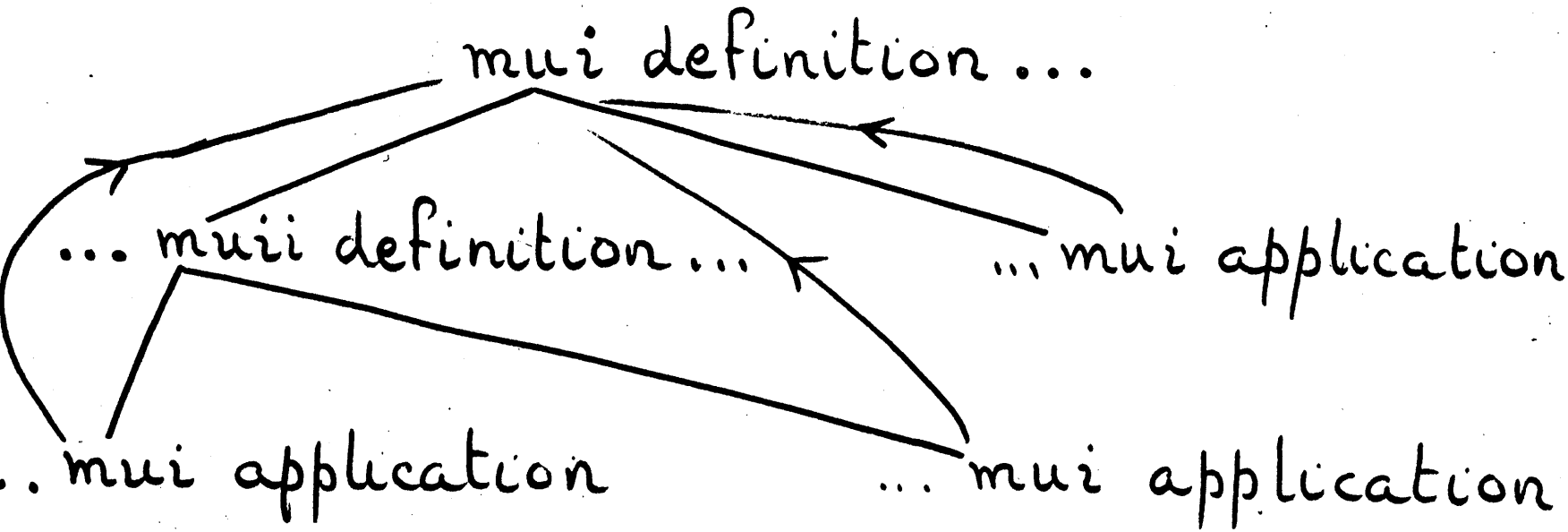
Recursive modes with finite spelling

```
MODE :: PLAIN; STOWED; REF to MODE;
   PROCEDURE; UNITED;
   MU definition of MODE; MU application.

MU ::      mu TALLY.
```

mui definition of structured with reference to muii definition of structured with reference to mui application field letter x reference to mui application field letter y mode field letter x reference to mui application field letter y mode

Mode spelling



Equivalence of modes

WHETHER SAFE1 MOID1 equivalent SAFE2 MOID2 :

- where ⟨SAFE1⟩ contains ⟨remember MOID1 MOID2⟩
  or ⟨SAFE2⟩ contains ⟨remember MOID2 MOID1⟩,
  WHETHER true ;
- unless ⟨SAFE1⟩ contains ⟨remember MOID1 MOID2⟩
  or ⟨SAFE2⟩ contains ⟨remember MOID2 MOID1⟩,
  WHETHER ⟨HEAD3⟩ is ⟨HEAD4⟩
  and remember MOID1 MOID2 SAFE3
  TAILETY3 equivalent SAFE4 TAILETY4,
  where SAFE3 HEAD3 TAILETY3 develops from SAFE1 MOID1
  and SAFE4 HEAD4 TAILETY4 develops from SAFE2 MOID2.

HEAD ::	TAILETY ::
PLAIN	EMPTY
PREF	MODE
structured with	FIELDS mode
FLEXETY ROWS of	MODE
procedure with	PARAMETERS yielding MOID
union of	MOODS mode
void	EMPTY

develops from

WHETHER SAFE2 HEAD TAILETY

develops from SAFE1 MOID :

- where <MOID> is <HEAD TAILETY>,
  WHETHER <HEAD> shields SAFE1 to SAFE2 ;
- where <MOID> is <MU definition of MODE>,
  unless <SAFE1> contains <MU has>,
  WHETHER SAFE2 HEAD TAILETY
  develops from MU has MODE SAFE1 MODE;
- where <MOID> is <MU application>
  and <SAFE1> is <NOTION MU has MODE SAFE3>
  and <NOTION> contains <yin>
  and <NOTION> contains <yang>,
  WHETHER SAFE2 HEAD TAILETY
  develops from SAFE1 MODE.