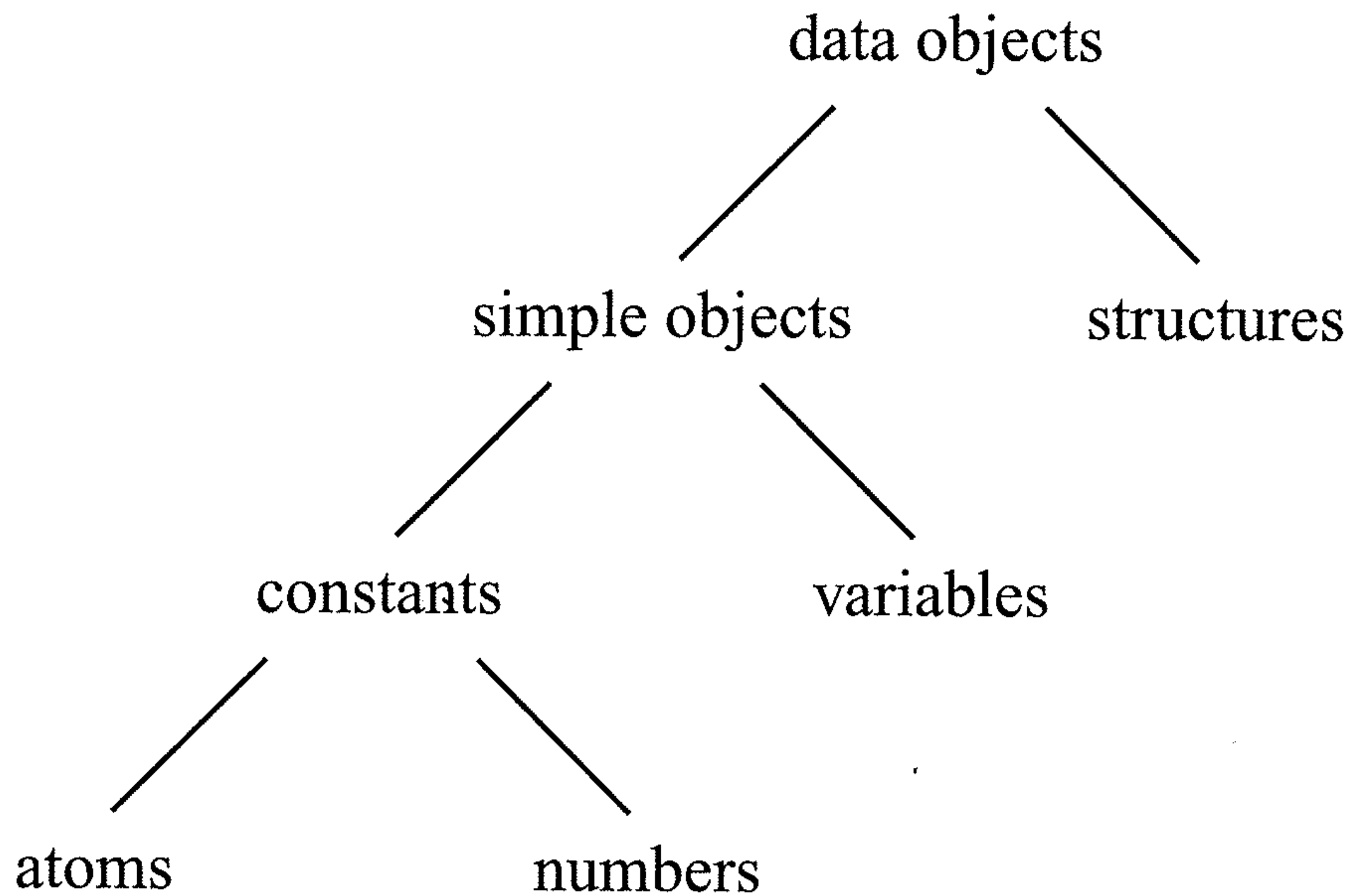

Figure 2.1 Data objects in Prolog.

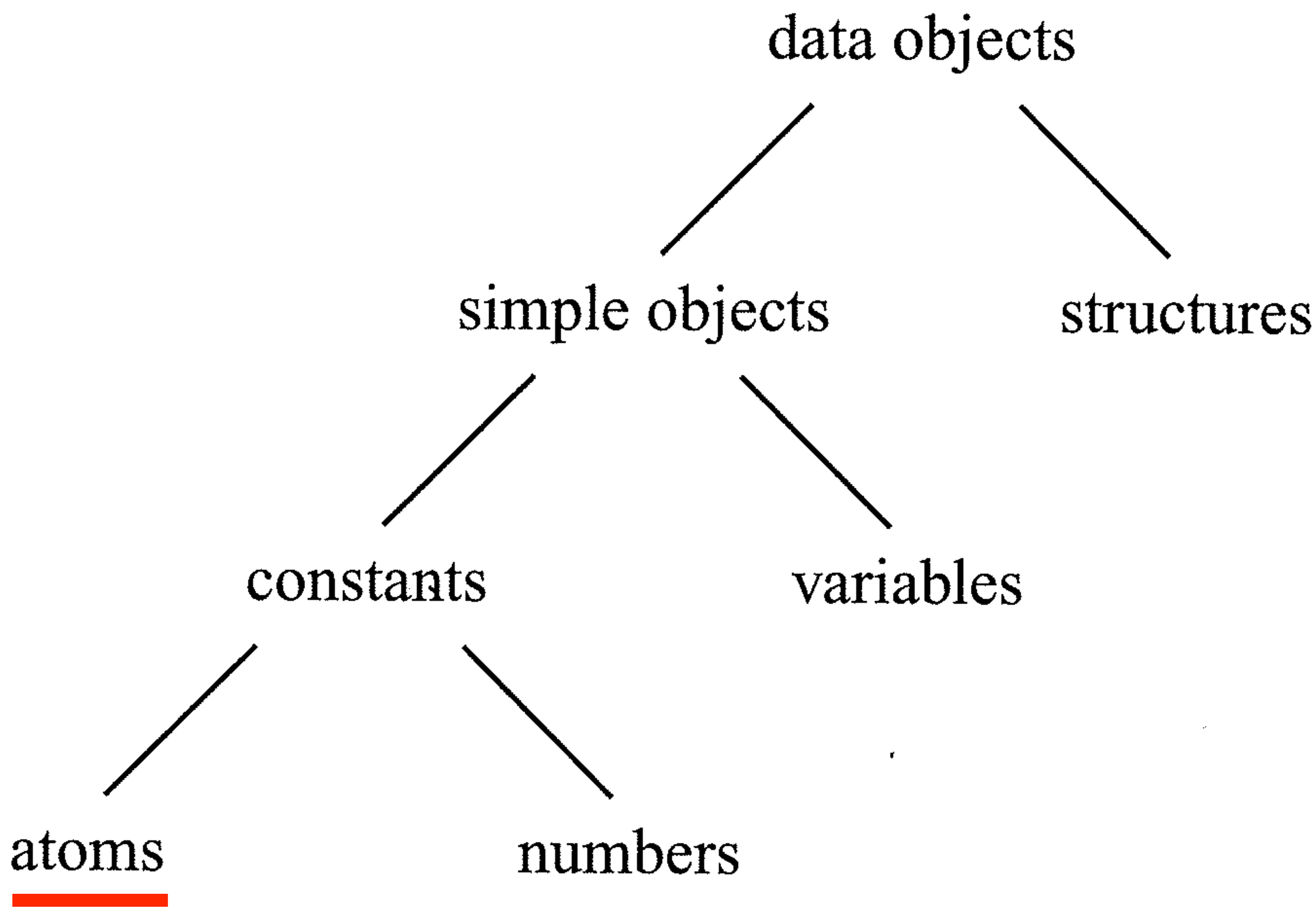

Figure 2.1 Data objects in Prolog.

Atoms

- Start with lowercase letters
- Strings of special characters, e.g. `:-` is an atom
- Enclosed in single quotes, e.g. `'Tom'` is an atom

Anonymous variables

- Singleton variable – variable in a rule that is named but not used
- Anonymous variable – an unnamed variable in a rule
- Avoid singleton variables

Structures

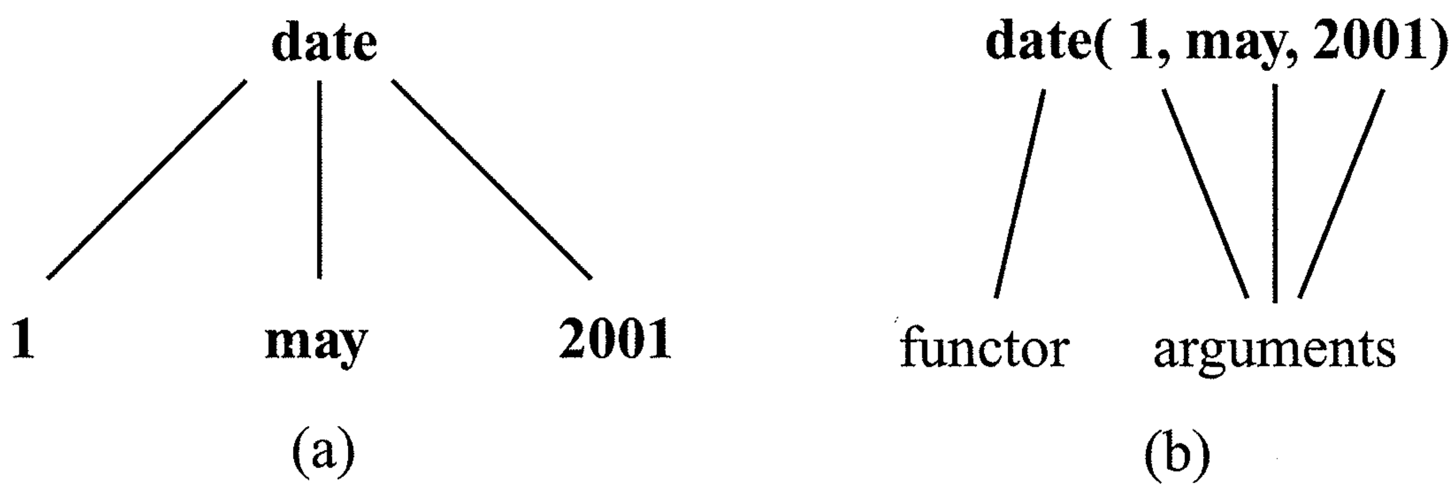

Figure 2.2 Date is an example of a structured object: (a) as it is represented as a tree; (b) as it is written in Prolog.

Structures

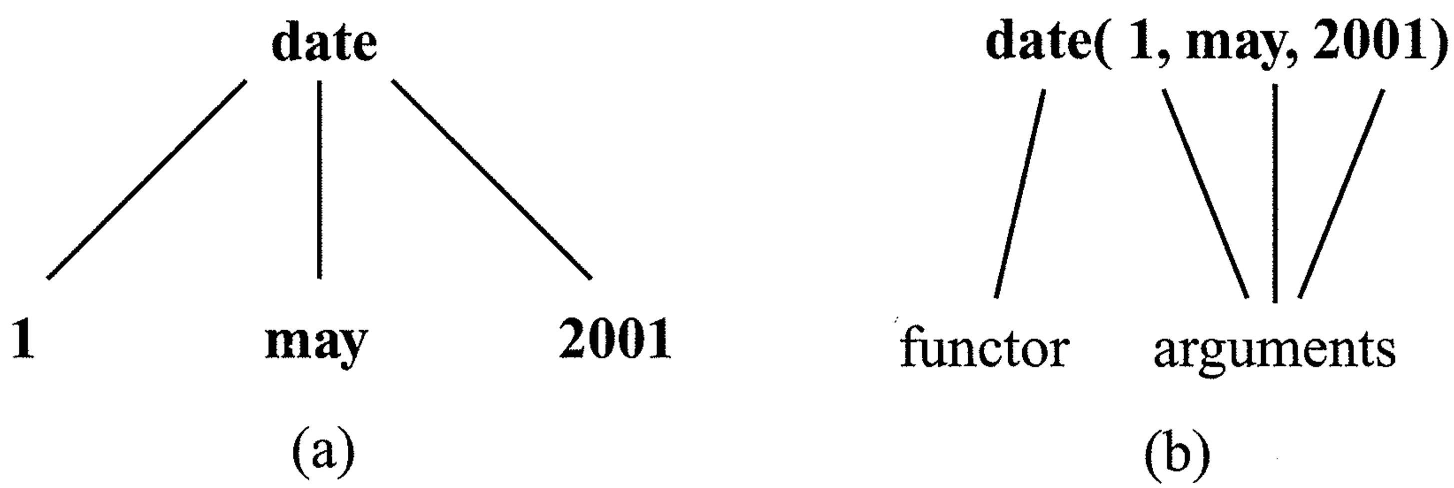

Figure 2.2 Date is an example of a structured object: (a) as it is represented as a tree; (b) as it is written in Prolog.

`date( 1, may, 2001)`

Name is `date`

Arity is 3

Terms

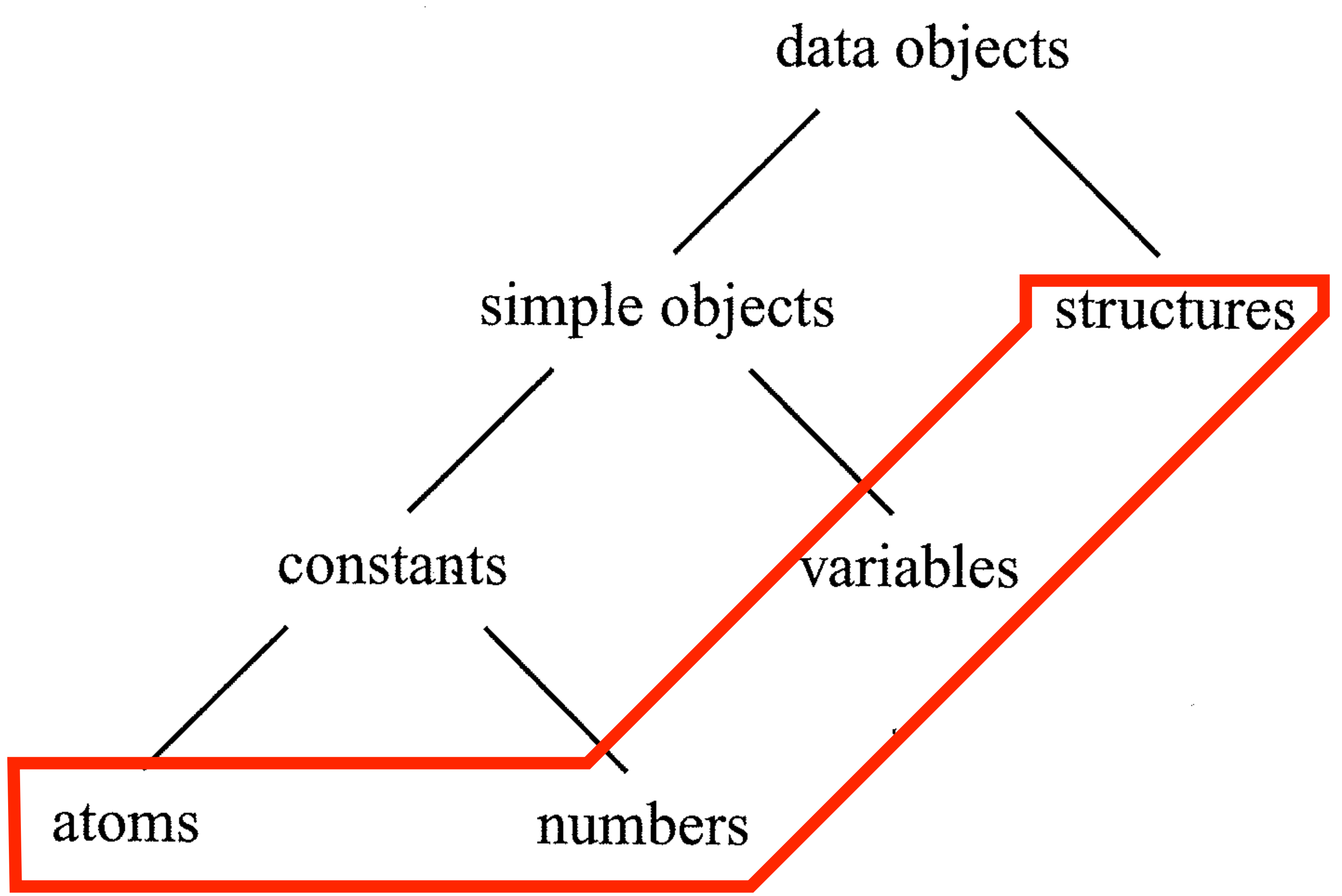

Figure 2.1 Data objects in Prolog.

Atoms, numbers, variables and structures are all terms.

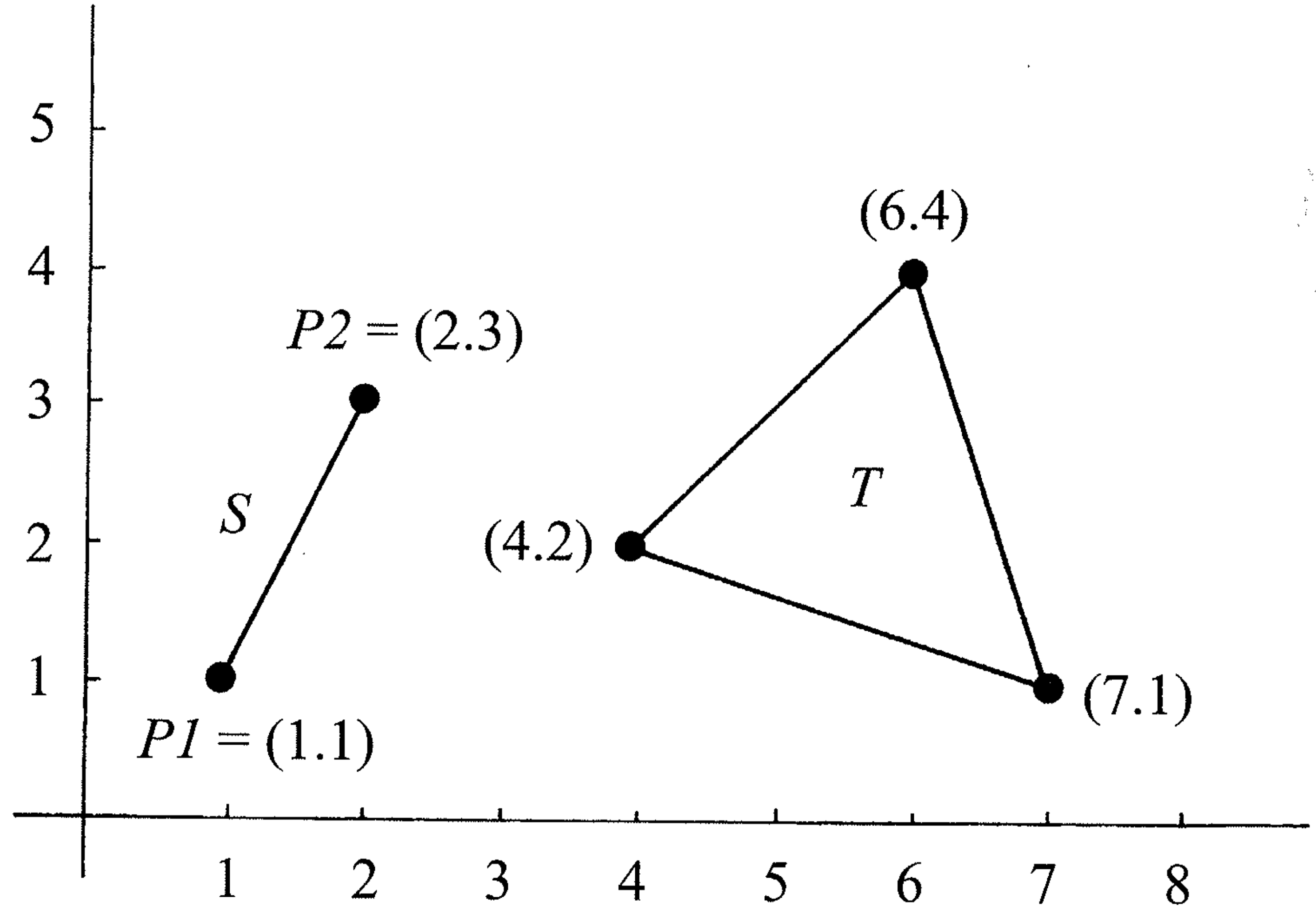

Figure 2.3 Some simple geometric objects.

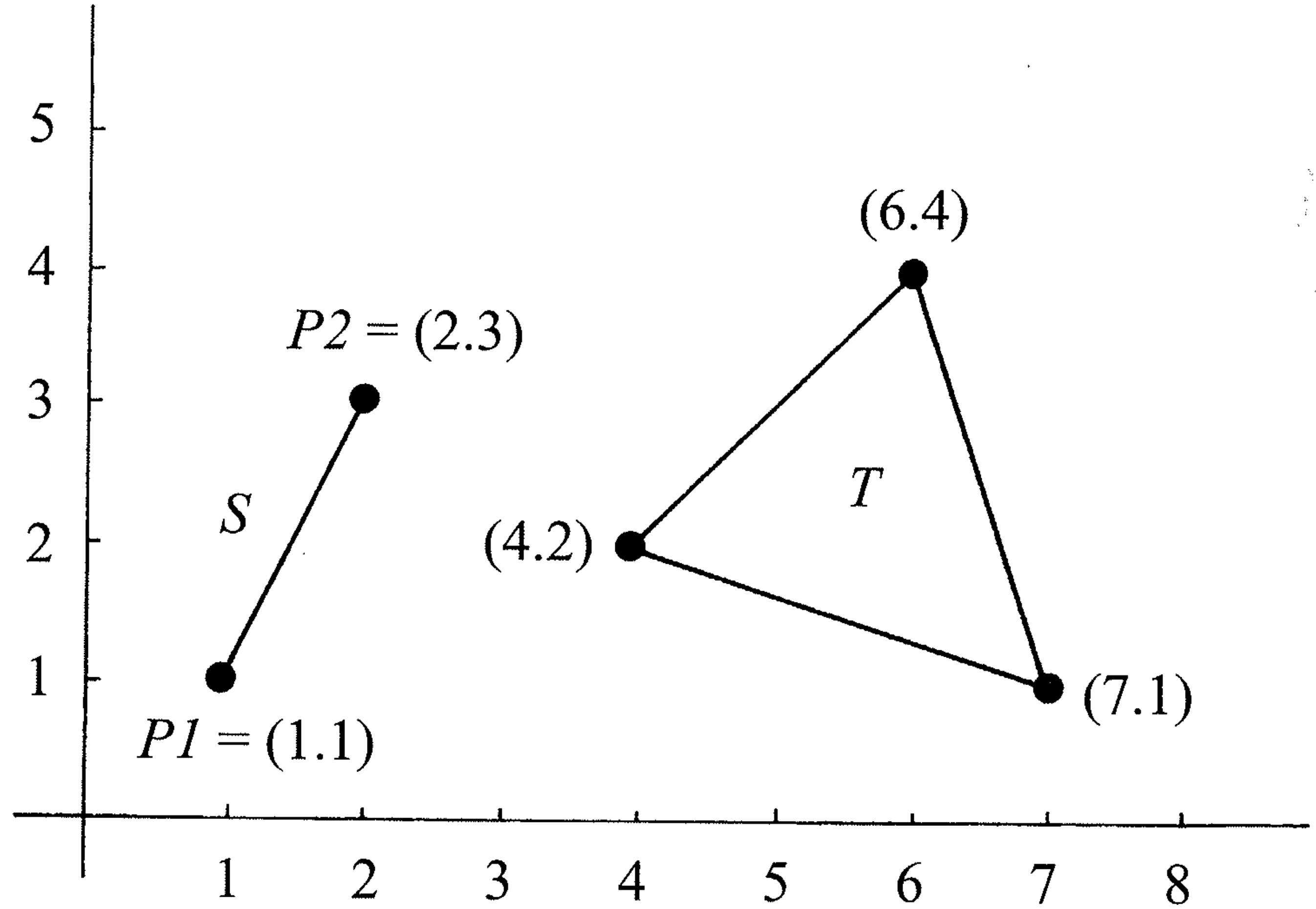

Figure 2.3 Some simple geometric objects.

P1 represented as `point( 1, 1).`

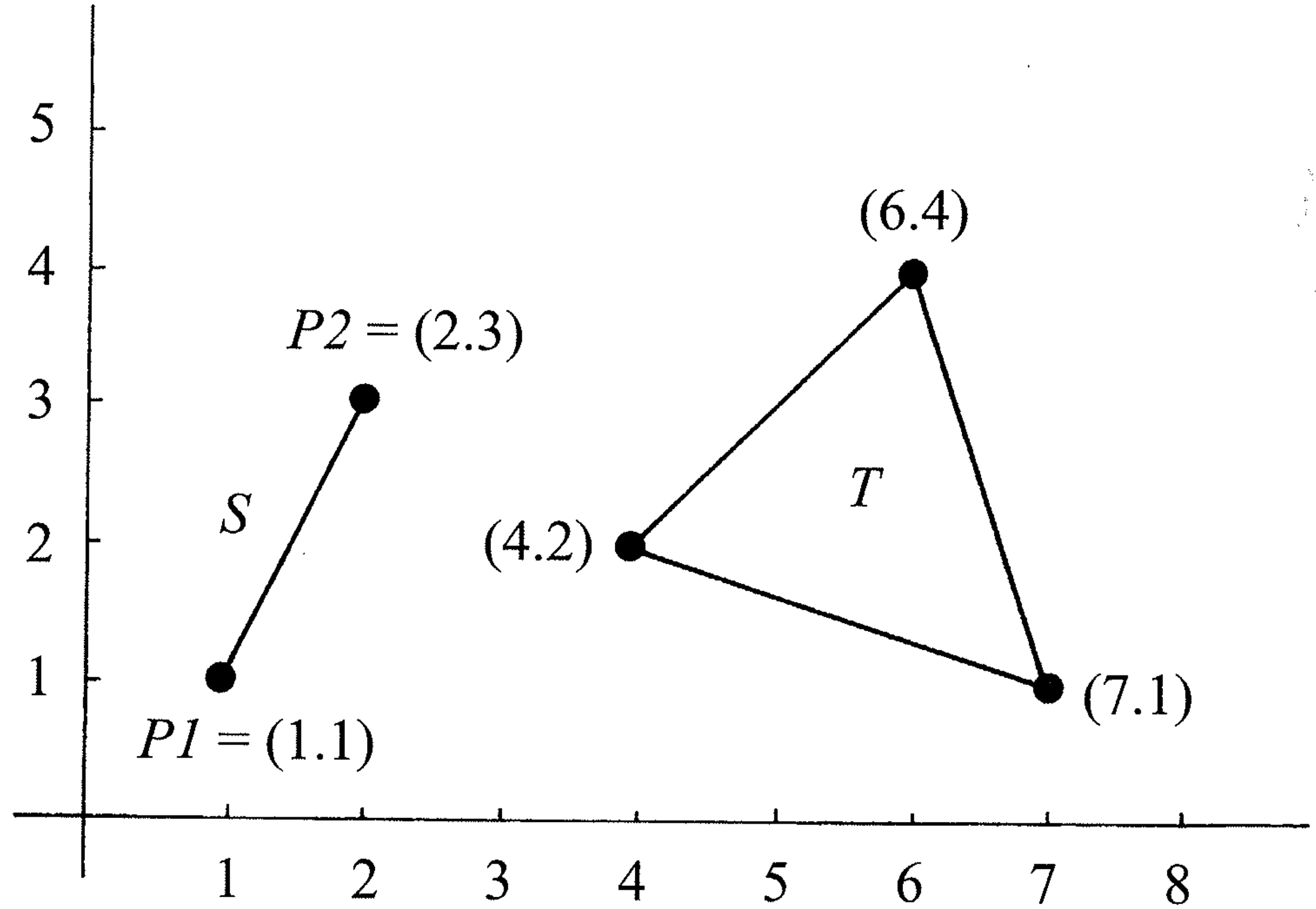

Figure 2.3 Some simple geometric objects.

P1 represented as `point( 1, 1).`

S represented as `seg( point( 1, 1), point( 2, 3).`

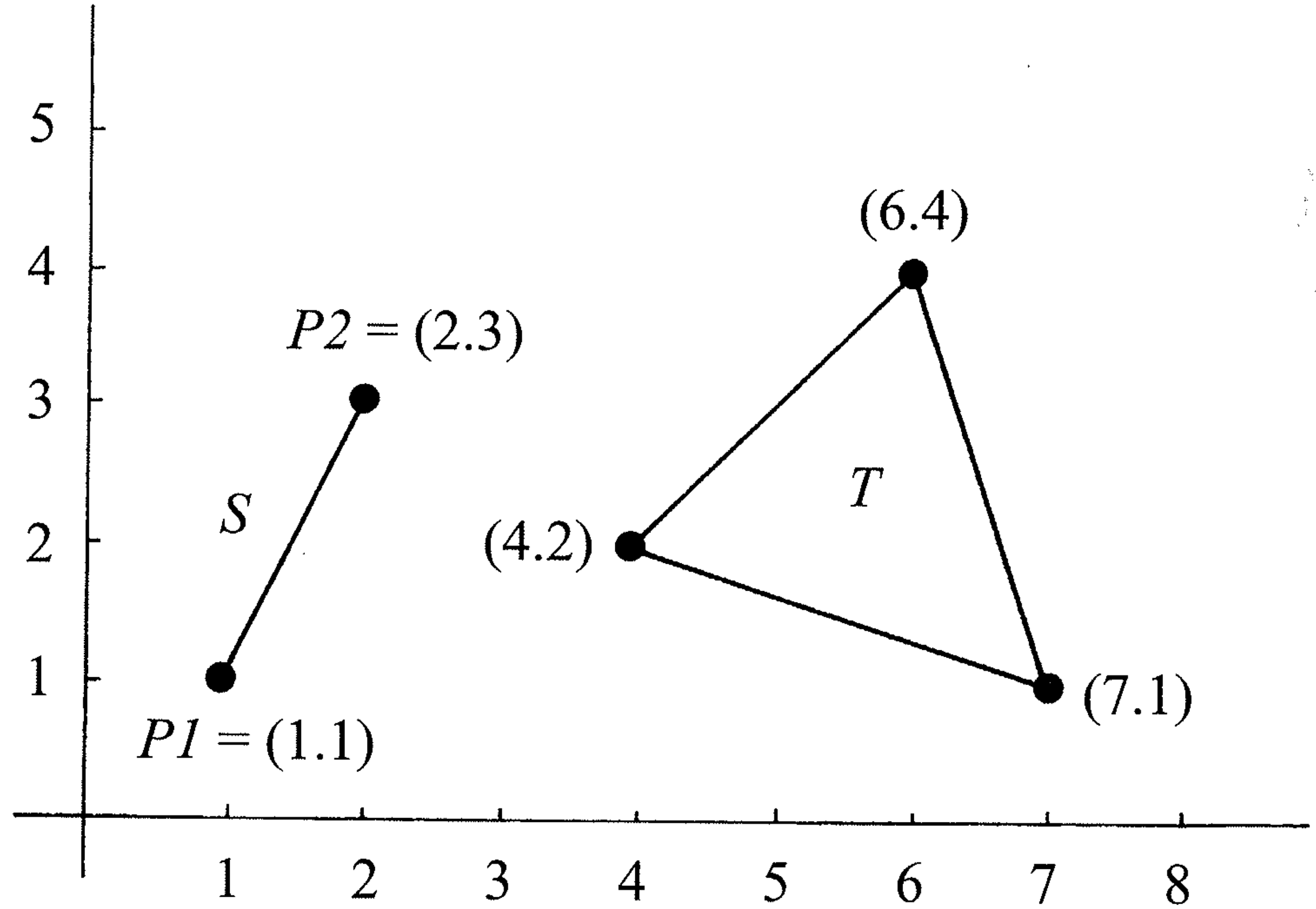

Figure 2.3 Some simple geometric objects.

P1 represented as `point( 1, 1).`
S represented as `seg( point( 1, 1), point( 2, 3).`
T represented as
`triangle( point( 4, 2), point( 6, 4), point( 7, 1)).`

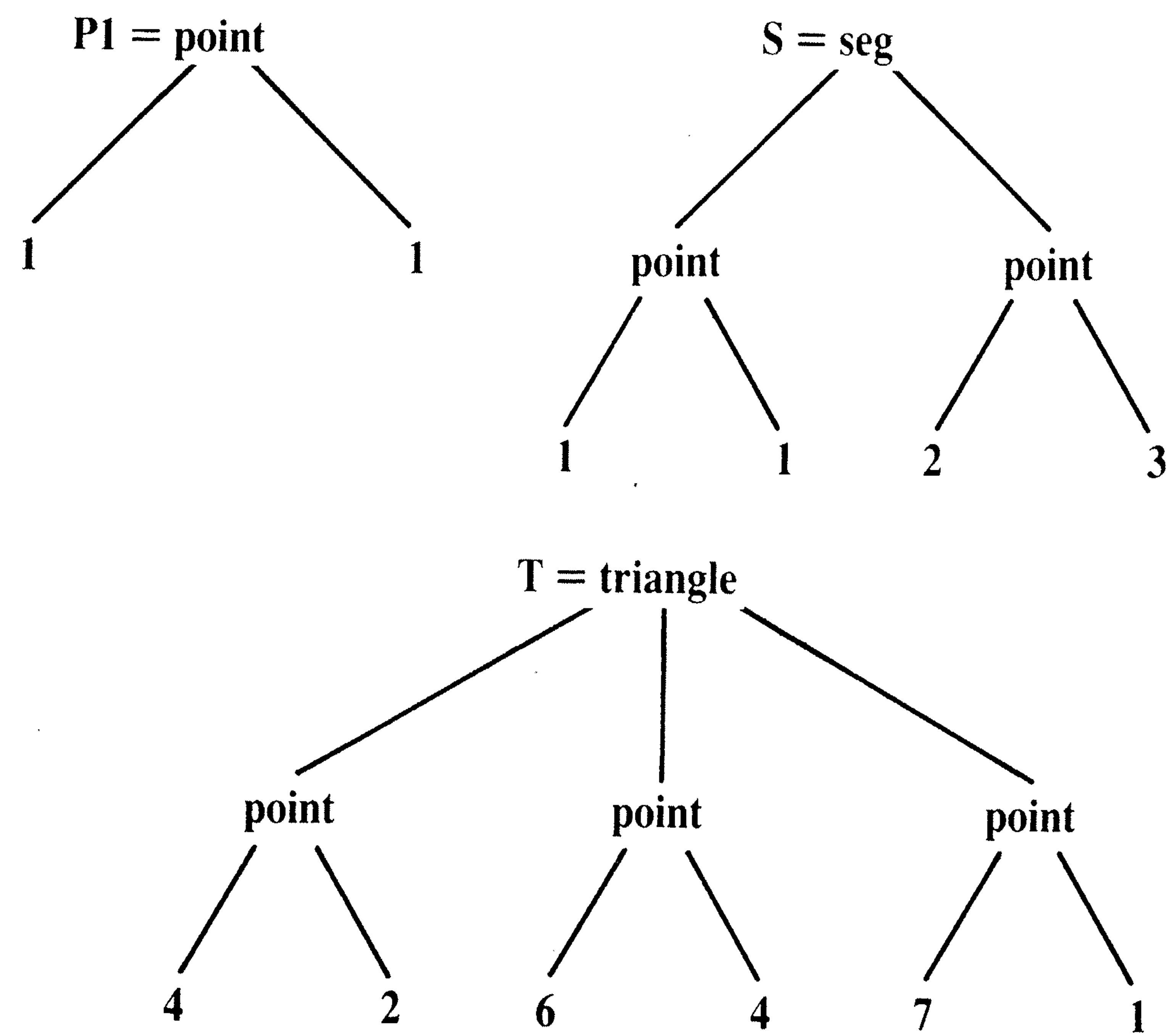

Figure 2.4 Tree representation of the objects in Figure 2.3.

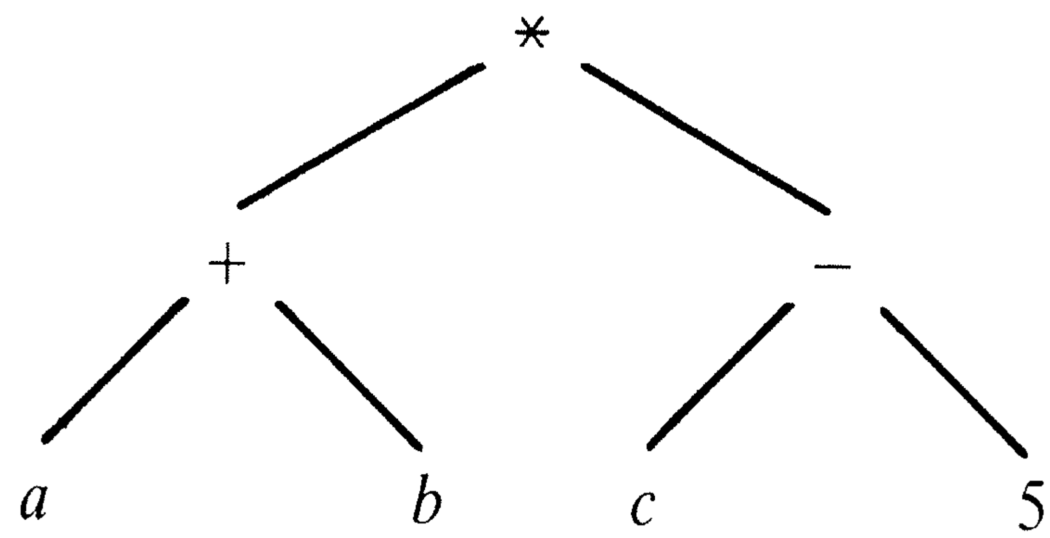

Figure 2.5 A tree structure that corresponds to the arithmetic expression $(a + b) * (c - 5)$.

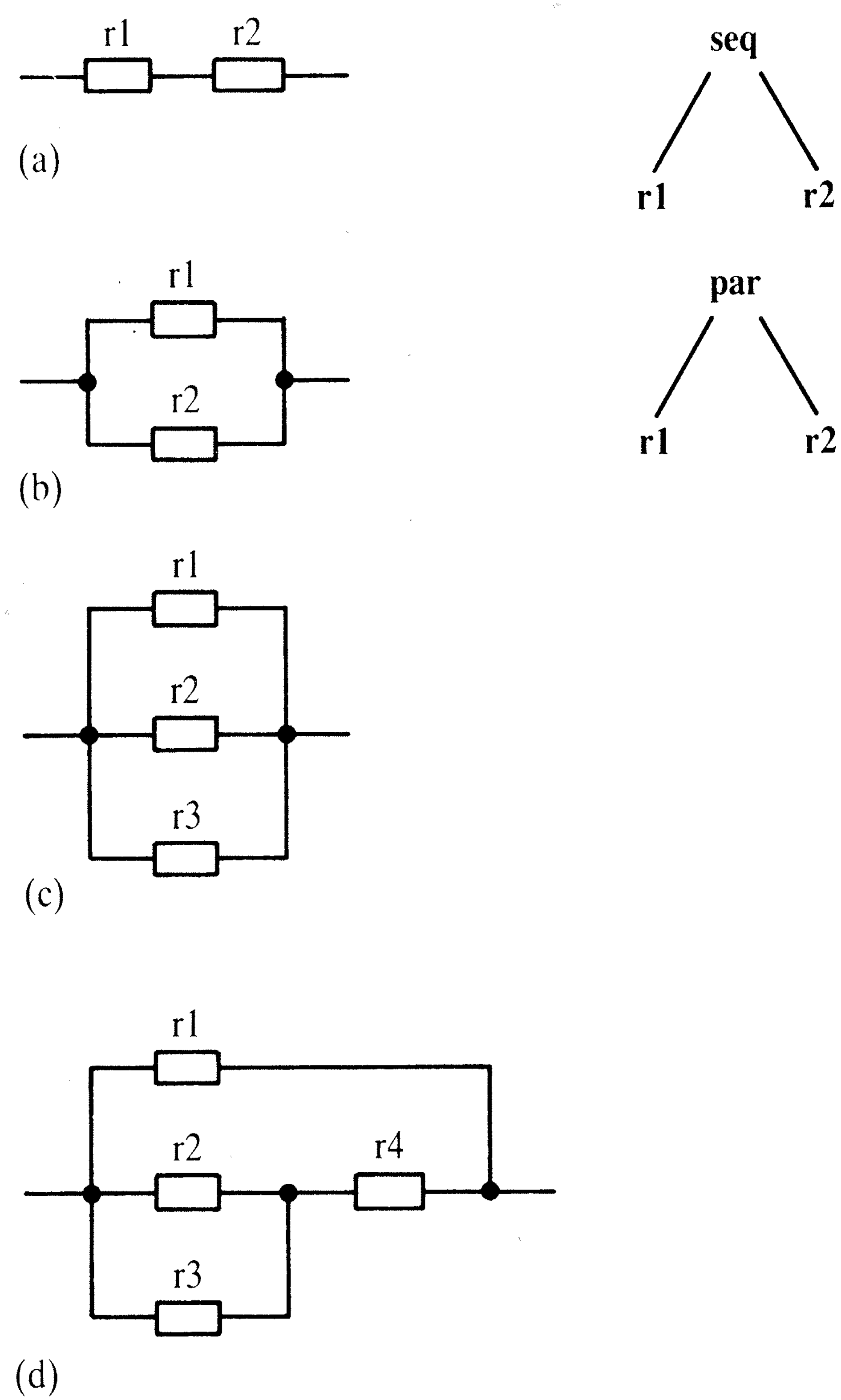

Figure 2.6 Some simple electric circuits and their tree representations: (a) sequential composition of resistors r1 and r2; (b) parallel composition of two resistors; (c) parallel composition of three resistors; (d) parallel composition of r1 and another circuit.

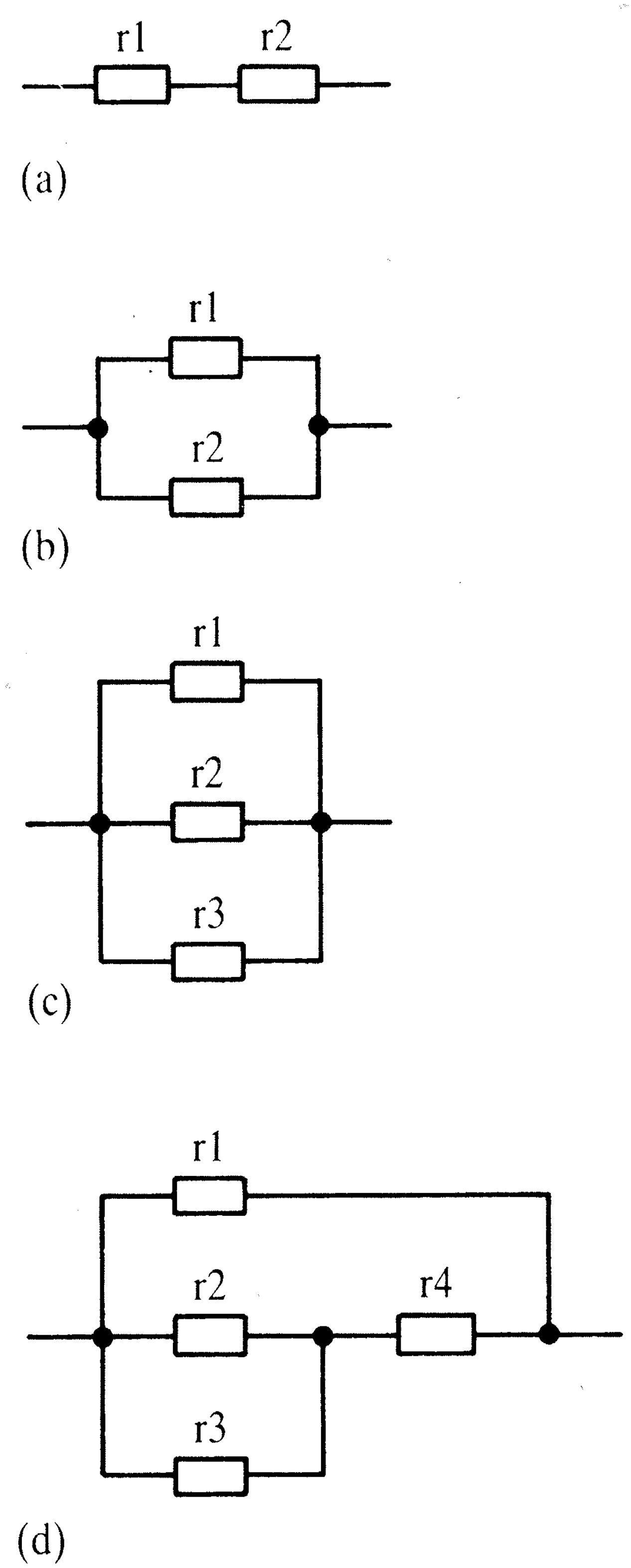

Figure 2.6 Some simple electric circuits and their tree representations: (a) sequential composition of resistors r1 and r2; (b) parallel composition of two resistors; (c) parallel composition of three resistors; (d) parallel composition of r1 and another circuit.

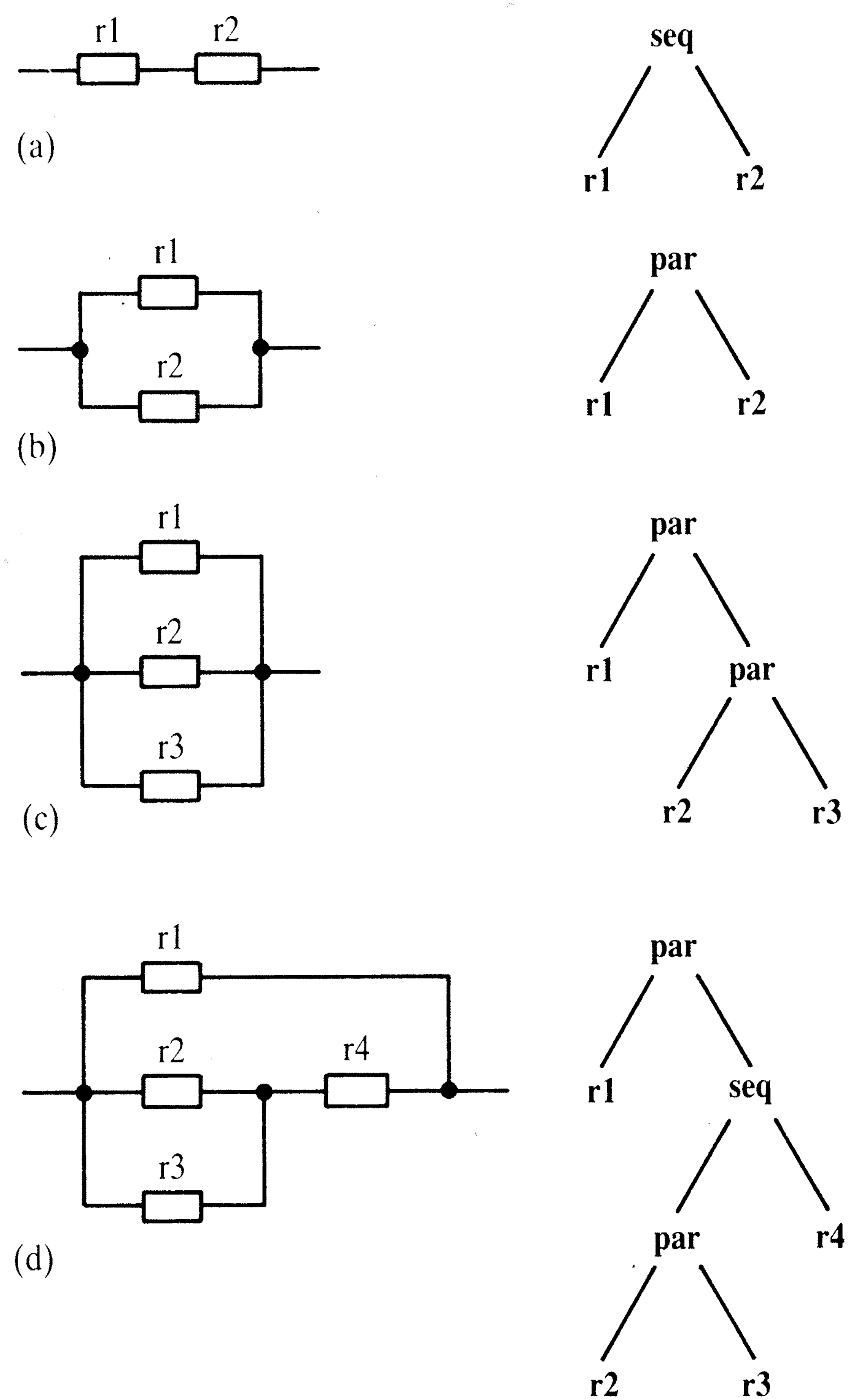

Figure 2.6 Some simple electric circuits and their tree representations: (a) sequential composition of resistors r1 and r2; (b) parallel composition of two resistors; (c) parallel composition of three resistors; (d) parallel composition of r1 and another circuit.

Matching
= is the matching operator

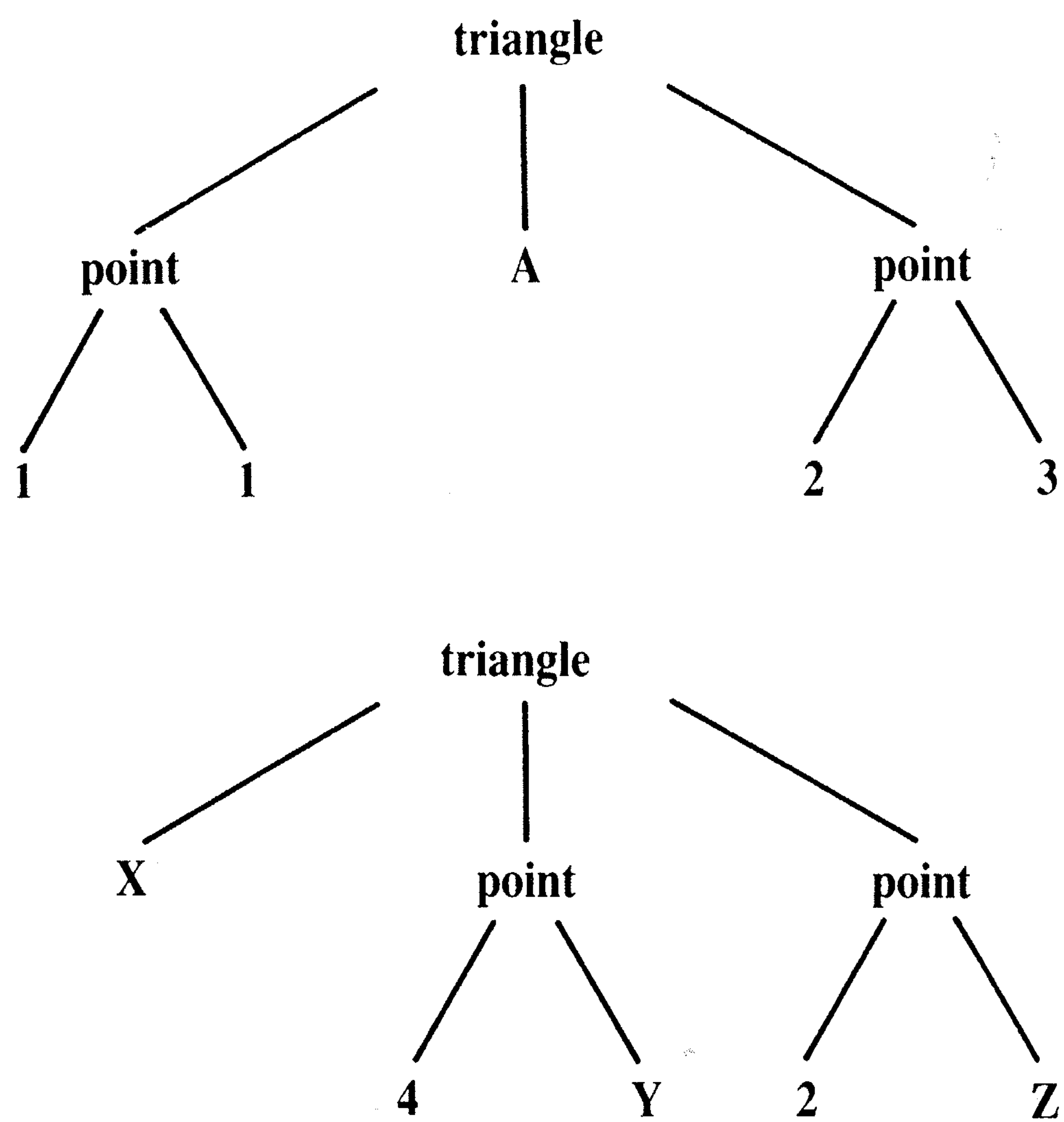

Figure 2.7 Matching **triangle(point(1,1), A, point(2,3)) = triangle(X, point(4,Y), point(2,Z))**

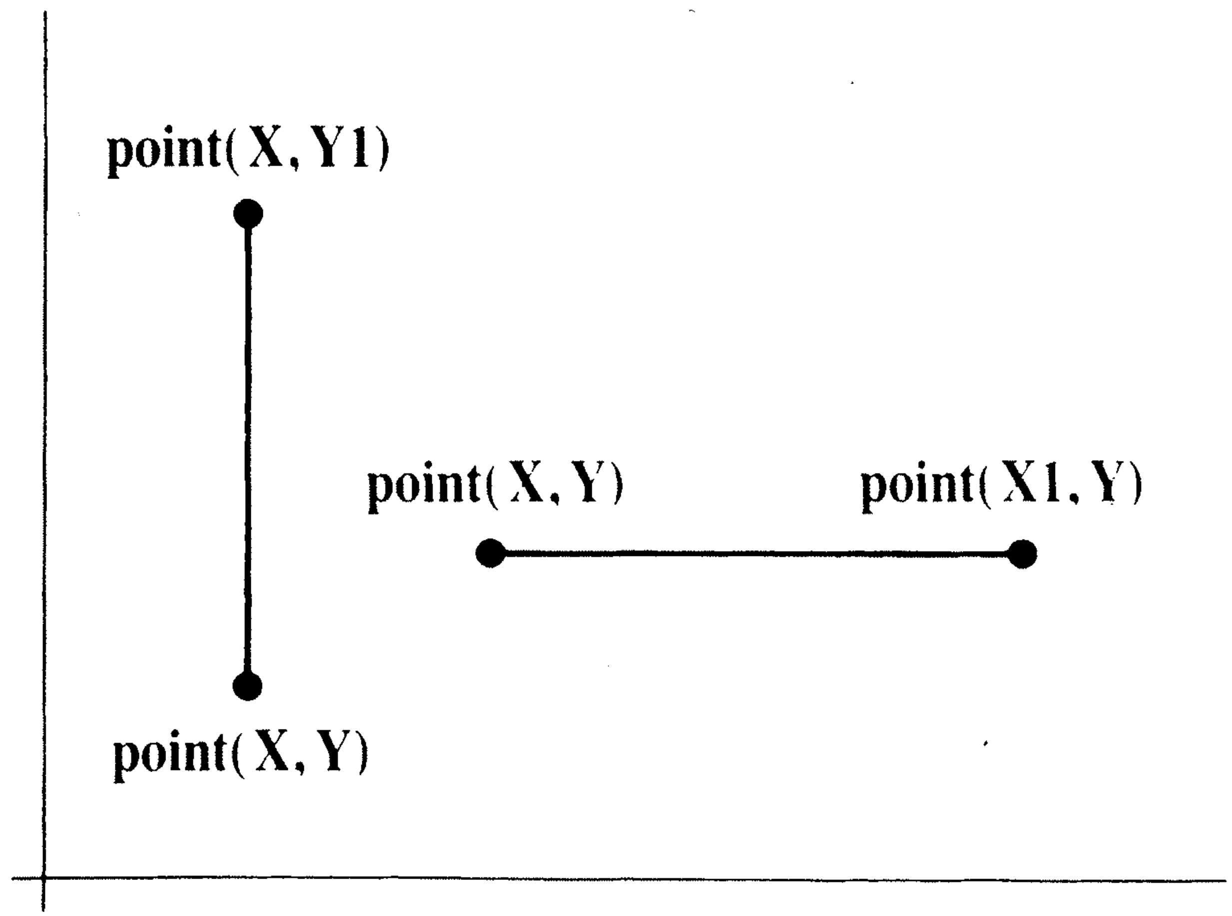

Figure 2.8 Illustration of vertical and horizontal line segments.

Matching
= is the matching operator

(1) If S and T are constants then S and T match only if they are the same object.

(2) If S is a variable and T is anything, then they match, and S is instantiated to T. Conversely, if T is a variable then T is instantiated to S.

(3) If S and T are structures then they match only if

  (a) S and T have the same principal functor, and

  (b) all their corresponding components match.

  The resulting instantiation is determined by the matching of the components.

Definitions

Clause – a rule or a fact

Instance of clause `C` – `C` with each variable substituted by some term.

`,` is conjunction "and"

`;` is disjunction "or"

```
P :- Q ; R.
```

is the same as

```
P :- Q.
P :- R.
```

Declarative meaning – what?

A goal G is true (that is, satisfiable, or logically follows from the program) if and only if:

(1) there is a clause C in the program such that

(2) there is a clause instance I of C such that

  (a) the head of I is identical to G, and

  (b) all the goals in the body of I are true.

Query, `?- G.`

head body

```
G :- Q, R, S.
```

goals

Procedural meaning – how?

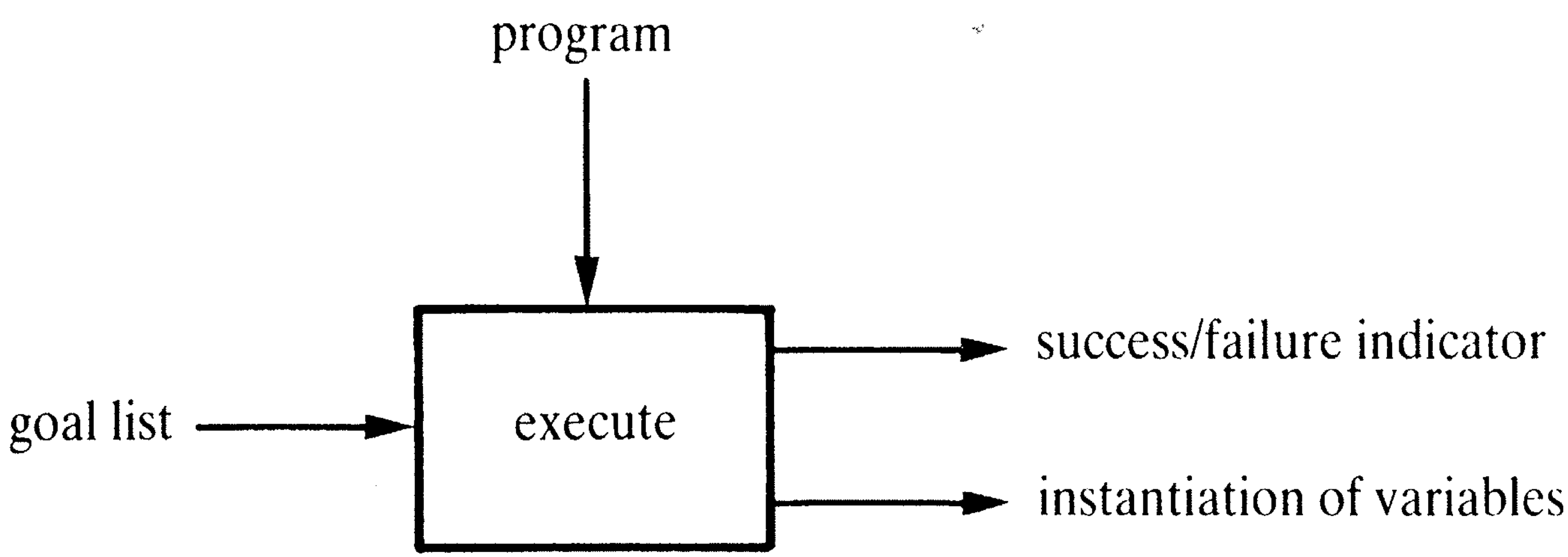

Figure 2.9 Input/output view of the procedure that executes a list of goals.

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
dark( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
dark( X), big( X).
```

Program (data base)

```
big( bear).          % Clause 1
big( elephant).      % Clause 2
small( cat).         % Clause 3

brown( bear).        % Clause 4
black( cat).         % Clause 5
gray( elephant).     % Clause 6
dark( Z) :-
  black( Z).         % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).         % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).        % Clause 1
big( elephant).    % Clause 2
small( cat).       % Clause 3

brown( bear).      % Clause 4
black( cat).       % Clause 5
gray( elephant).   % Clause 6
dark( Z) :-
  black( Z).       % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).       % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).        % Clause 1
big( elephant).    % Clause 2
small( cat).       % Clause 3

brown( bear).      % Clause 4
black( cat).       % Clause 5
gray( elephant).   % Clause 6
dark( Z) :-
  black( Z).       % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).       % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).       % Clause 1
big( elephant).   % Clause 2
small( cat).      % Clause 3

brown( bear).     % Clause 4
black( cat).      % Clause 5
gray( elephant).  % Clause 6
dark( Z) :-
  black( Z).      % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).      % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( cat).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( cat).
```

Program (data base)

```
big( bear).        % Clause 1
big( elephant).    % Clause 2
small( cat).       % Clause 3

brown( bear).      % Clause 4
black( cat).       % Clause 5
gray( elephant).   % Clause 6
dark( Z) :-
  black( Z).       % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).       % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( cat).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

→ Fail

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( cat).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Backtrack to `black( cat).`

Program (data base)

```
big( bear).          % Clause 1
big( elephant).      % Clause 2
small( cat).         % Clause 3

brown( bear).        % Clause 4
black( cat).         % Clause 5
gray( elephant).     % Clause 6
dark( Z) :-
  black( Z).         % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).         % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Fail

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
black( X), big( X).
```

Backtrack to `dark( Z).`

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
dark( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
dark( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
brown( X), big( X).
```

Program (data base)

```
big( bear).          % Clause 1
big( elephant).      % Clause 2
small( cat).         % Clause 3

brown( bear).        % Clause 4
black( cat).         % Clause 5
gray( elephant).     % Clause 6
dark( Z) :-
  black( Z).         % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).         % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
brown( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
brown( X), big( X).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( bear).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
big( bear).
```

Program (data base)

```
big( bear).         % Clause 1
big( elephant).     % Clause 2
small( cat).        % Clause 3

brown( bear).       % Clause 4
black( cat).        % Clause 5
gray( elephant).    % Clause 6
dark( Z) :-
  black( Z).        % Clause 7: Anything black is dark
dark( Z) :-
  brown( Z).        % Clause 8: Anything brown is dark
```

Goal list

```
?- dark( X), big( X).
```

Execution trace

```
X = bear
```

procedure *execute* (*Program, GoalList, Success*);

Input arguments:

Program: list of clauses

GoalList: list of goals

Output argument:

Success: truth value; *Success* will become true if *GoalList* is true with respect to *Program*

Local variables:

Goal: goal

OtherGoals: list of goals

Satisfied: truth value

MatchOK: truth value

Instant: instantiation of variables

$H, H', B1, B1', \ldots, Bn, Bn'$: goals

Auxiliary functions:

empty(L): returns true if *L* is the empty list

head(L): returns the first element of list *L*

tail(L): returns the rest of *L*

append(L1,L2): appends list *L2* at the end of list *L1*

match(T1,T2,MatchOK,Instant): tries to match terms *T1* and *T2*; if succeeds then *MatchOK* is true and *Instant* is the corresponding instantiation of variables

substitute(Instant, Goals): substitutes variables in *Goals* according to instantiation *Instant*

```
begin
  if empty(GoalList) then Success := true
  else
    begin
      Goal := head(GoalList);
      OtherGoals := tail(GoalList);
      Satisfied := false;
      while not Satisfied and "more clauses in program" do
        begin
          Let next clause in Program be
            H :- B1, . . . , Bn.
          Construct a variant of this clause
            H' :- B1', . . . , Bn'.
          match(Goal,H',MatchOK,Instant);
          if MatchOK then
            begin
              NewGoals := append([B1', . . . ,Bn'], OtherGoals);
              NewGoals := substitute(Instant,NewGoals);
              execute(Program,NewGoals,Satisfied)
            end
        end;
      Success := Satisfied
    end
end;
```

Reordering clauses and goals

- Reordering can have a big effect on efficiency.

- In extreme cases, reordering can cause an infinite recursive loop.

```
anc2( X, Z):-
   parent( X, Y),
   anc2( Y, Z).

anc2( X, Z):-
   parent( X, Z).
```

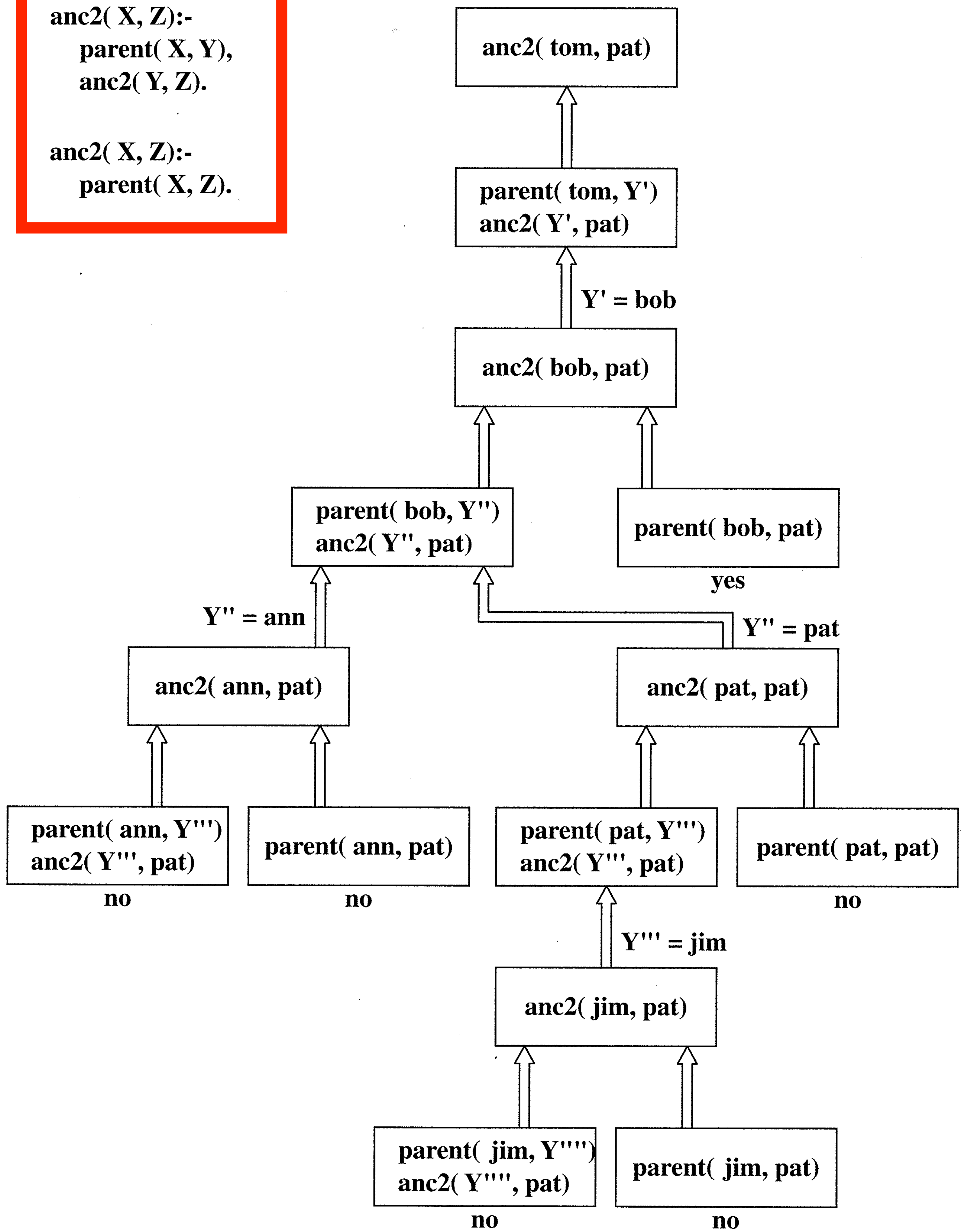

Figure 2.13 The complete execution trace to satisfy the goal **anc2(tom, pat)**. All the alternative paths in the large left subtree fail, before the right-most path succeeds.

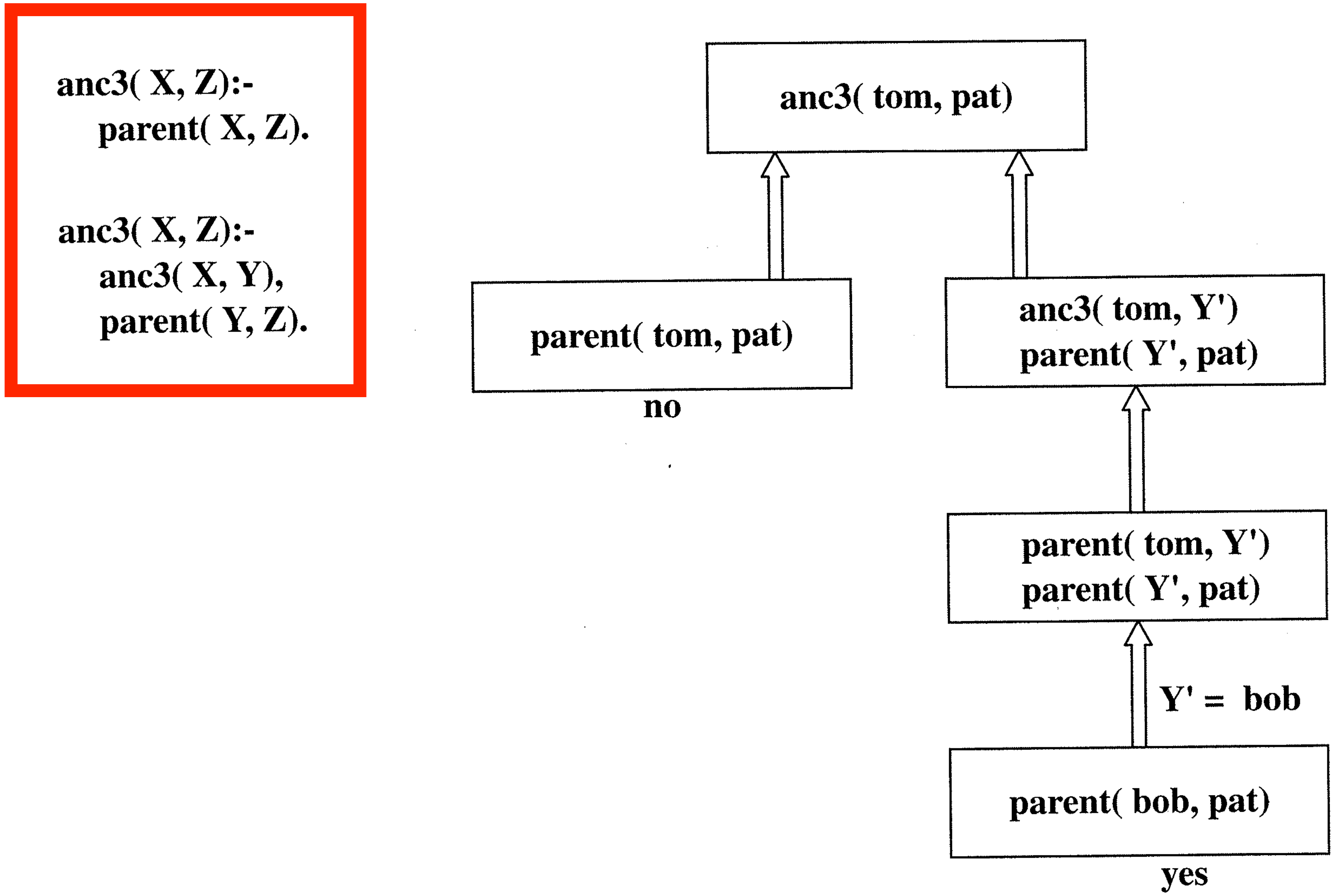

Figure 2.14 The execution trace to satisfy the goal **anc3(tom, pat)**.

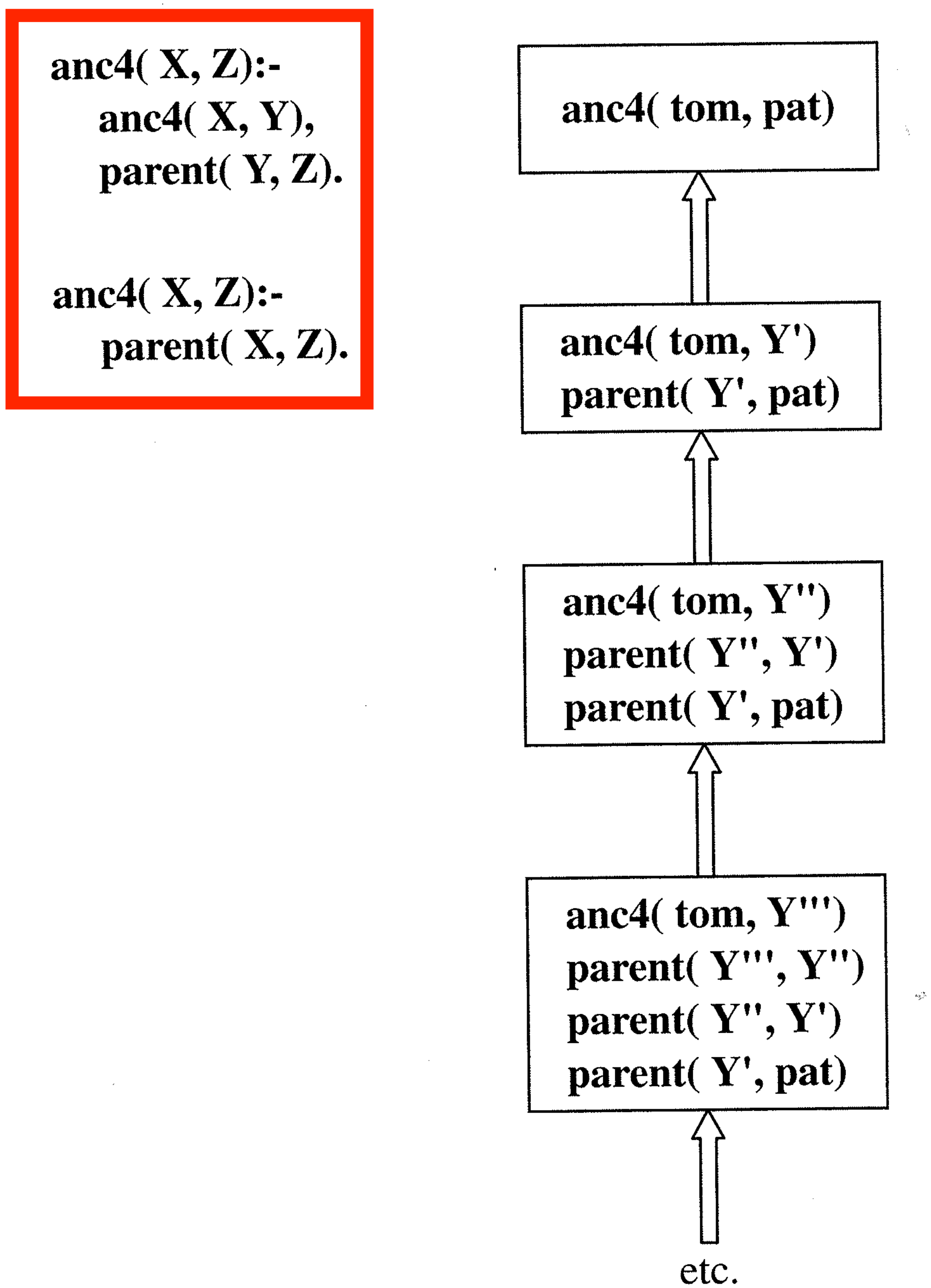

Figure 2.15 Infinite execution trace to satisfy the goal **anc4(tom, pat)**.