No Exit

PANAGIA
SLOV

CLOSED
DUE TO
HIGH WINDS

Type-tagging with Forth

Formulating Type Tagged Parse Trees as Forth Programs.

Dr Campbell Ritchie and Dr Bill Stoddart

Formal Methods and Programming Research Group
Teesside University, UK

A ::= A + T | A - T | T

A = A "+" T, A "-" T, T

A = A “ +” T, A “ -” T, T

T = T “ *” F, T “ /” F, F

F = N, I, “ (” A “)”

…

$$P_{\text{A}}(\text{“ } a + t\text{”}) = P_{\text{A}}(a)\ P_{\text{T}}(t)\ \text{“ +”}$$

$$P_{\text{T}}(\text{“ } t * f\text{”}) = P_{\text{T}}(t)\ P_{\text{F}}(f)\ \text{“ *”}$$

$$P_{\text{F}}(\text{“ } (a)\text{”}) = P_{\text{A}}(a)$$

$$P_{\text{F}}(\text{“ 123”}) = P_{\text{N}}(\text{“ 123”})$$

a + b * (c + d)

Becomes

a b c d + * +

$$\underline{a + b * (c + d)}$$

$$a\ \underline{b * (c + d)}\ +$$

$$a\ b\ \underline{(c + d)}\ *\ +$$

$$a\ b\ \underline{c + d}\ *\ +$$

$$a\ b\ c\ d\ +\ *\ +$$

1 + 2.5

“ 1” “ INT” “ 2.5” “ FLOAT” +_

“ 1 FLOAT 2.5 F+”

Examples of types of Expressions:

No 1:
{ 1, 2, 3 } is a member of the
“Set of INTs” which is expressed in
postfix as
“ INT SET”
and the expression in postfix is
“ INT { 1 , 2 , 3 ,} INT SET”

Examples of types of Expressions:

No 2:
1 $\mapsto$ “ foo”
is a member of INTs paired to STRINGs,
which can be expressed in postfix as
“ INT STRING PAIR”
and the expression in postfix is
“ 1 “ foo” $\mapsto$”

E = E “ $\mapsto$” E0, E0

“ $x \mapsto y$”

“ x” “ foo” “ y” “ bar” $\mapsto$_

“ x y $\mapsto$” “ foo bar PAIR”

```
: ↦_
(
     l-value:$ l-type:$ r-value:$ r-type:$ --
     az1 =  values catenated with ↦
     az2 = types catenated with PAIR
)
(: VALUE l-value VALUE l-type VALUE r-value VALUE r-type :)
l-value r-value AZ^     "   ↦" AZ^
l-type r-type AZ^     "    PAIR" AZ^
2LEAVE ;
```

The ,_ operation is used for creating lists for the contents of sets, etc.

It needs to check that the two types are *null* or the same type.

$S = S_1$ “ $\triangleleft$” S, S_1

$\mathbb{P}(T) \quad \triangleleft\ \mathbb{P}(T \times U) \qquad\qquad \rightarrow \mathbb{P}(T \times U)$
Set of T $\triangleleft$ Set of T paired to U

$a \triangleleft b$

“ *a*” “ foo SET” “ *b*” “foo bar PAIR SET” $\triangleleft$_

“ *a b* $\triangleleft$” “ foo bar PAIR SET”

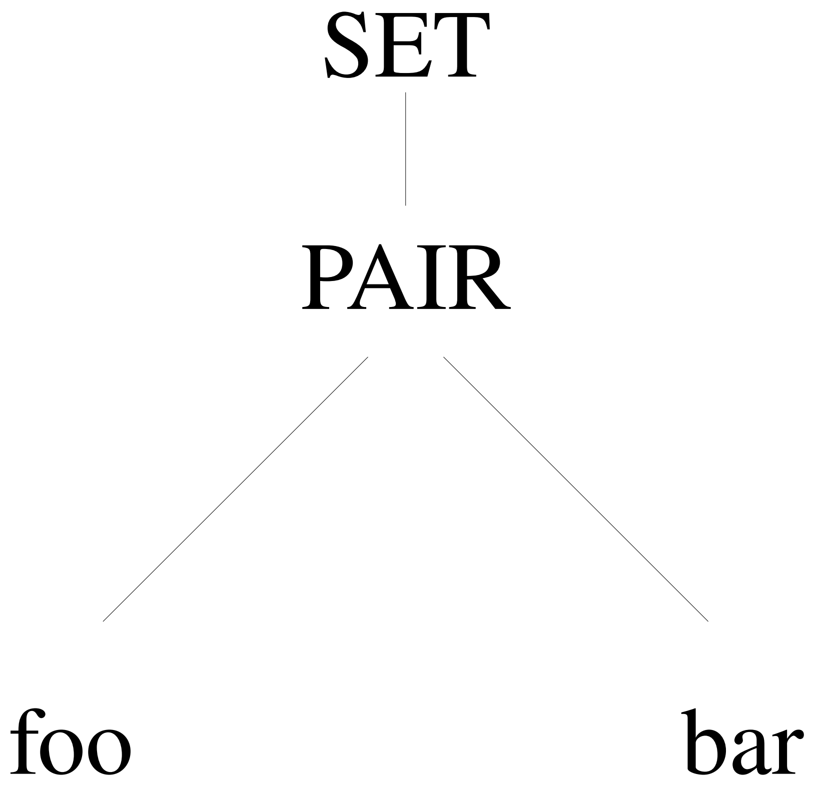
“ foo bar PAIR SET”

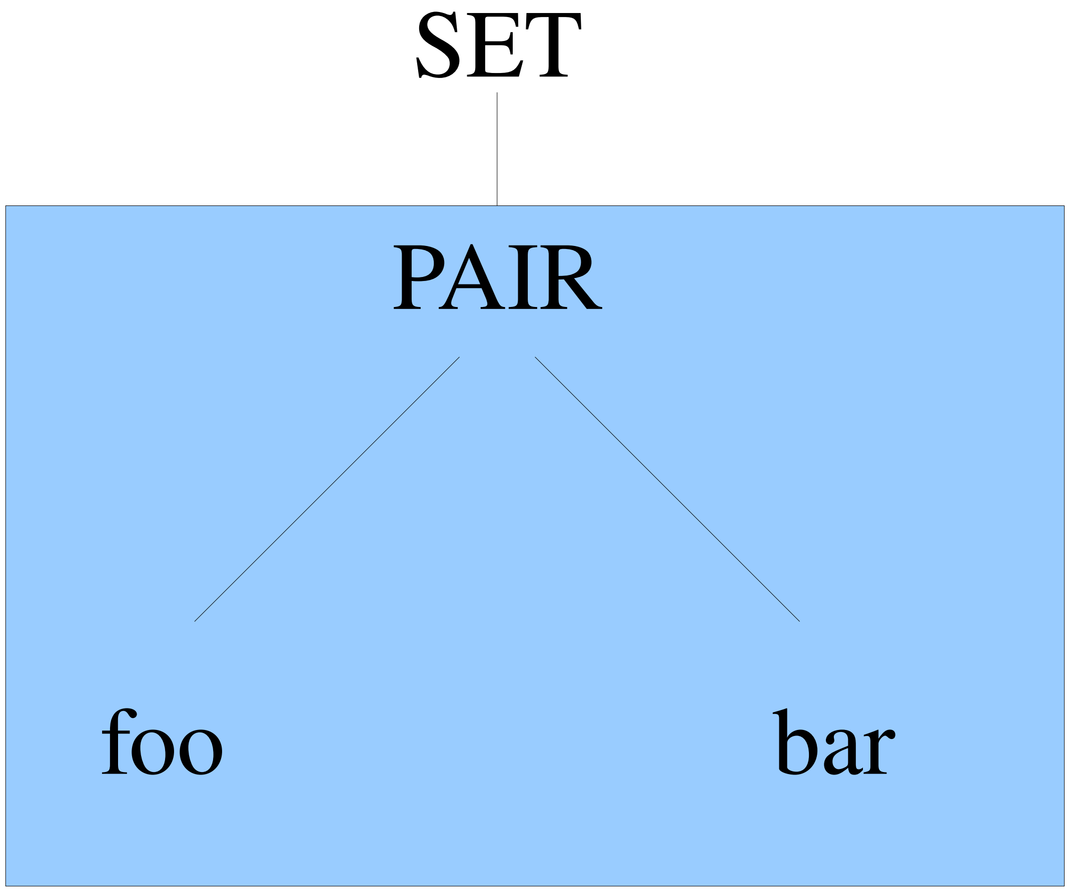
“ foo SET”

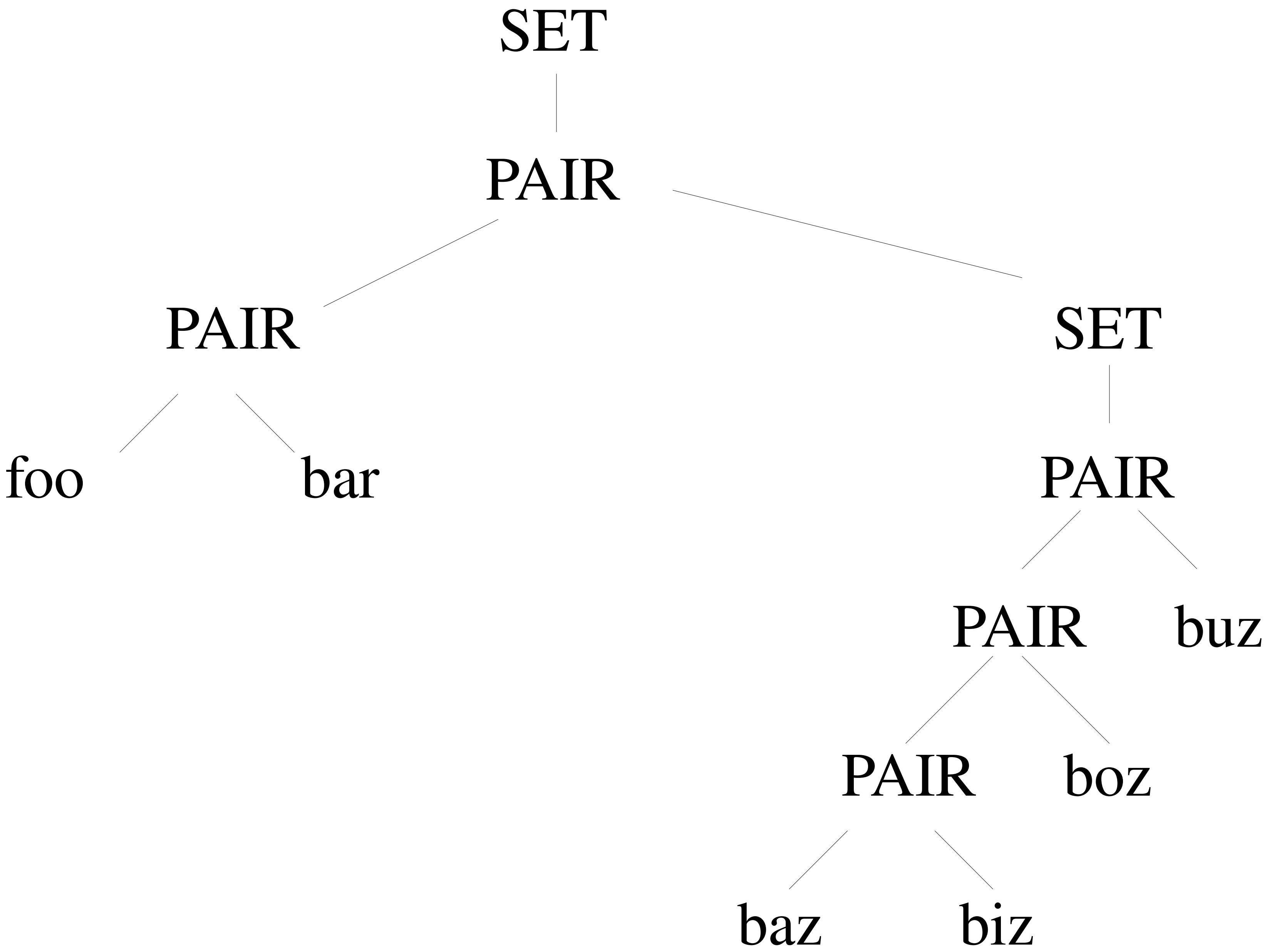
“ foo bar PAIR SET”

“ foo SET”

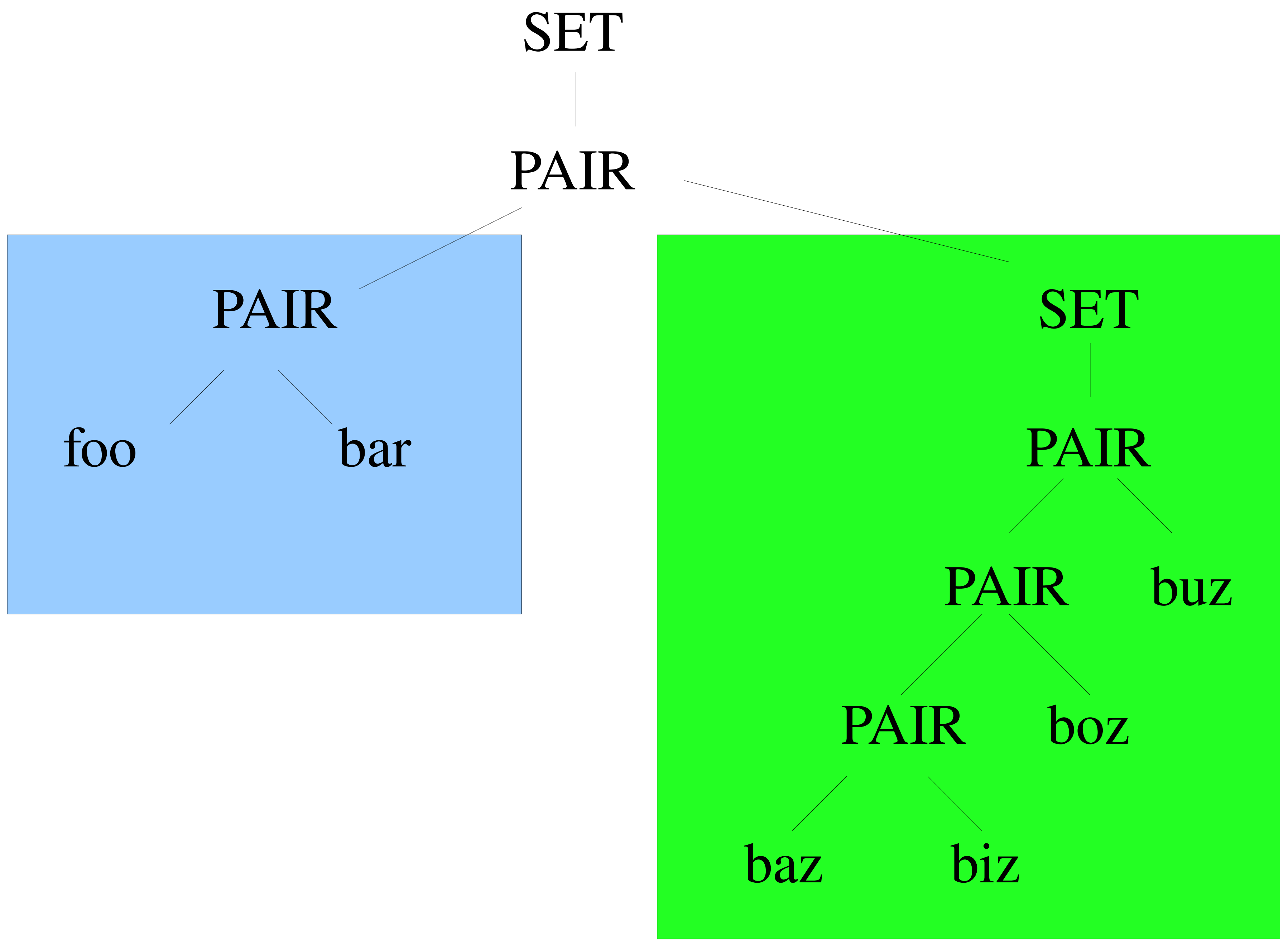
“ *a*” “ foo bar PAIR SET” “ *b*”
“ foo bar PAIR baz biz PAIR boz PAIR buz
PAIR SET PAIR SET”
$\triangleleft$_


SET
PAIR
foo
bar

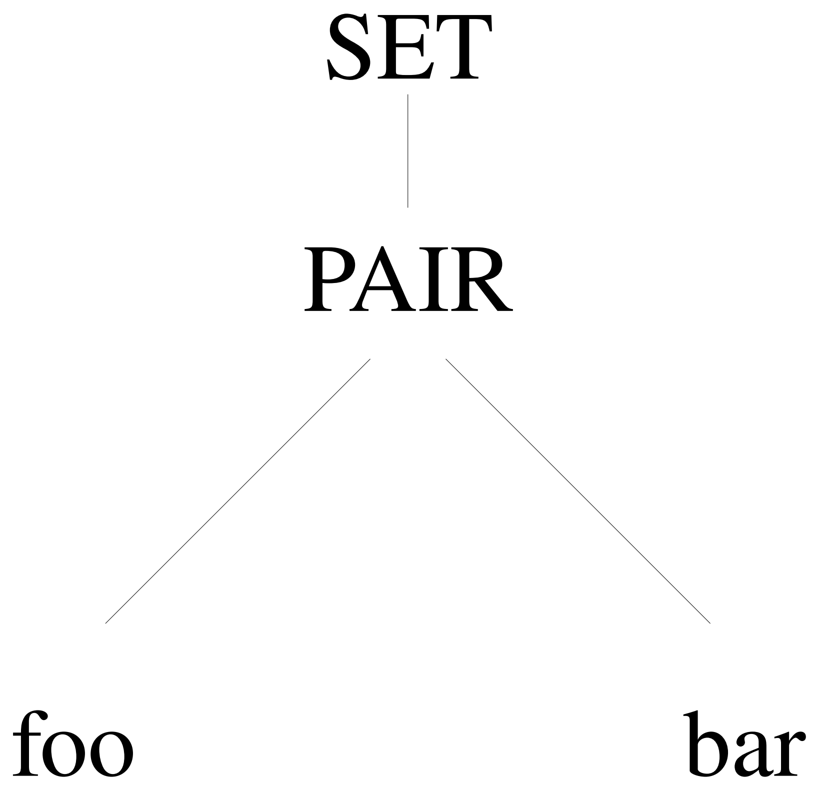

SET
PAIR
foo
bar

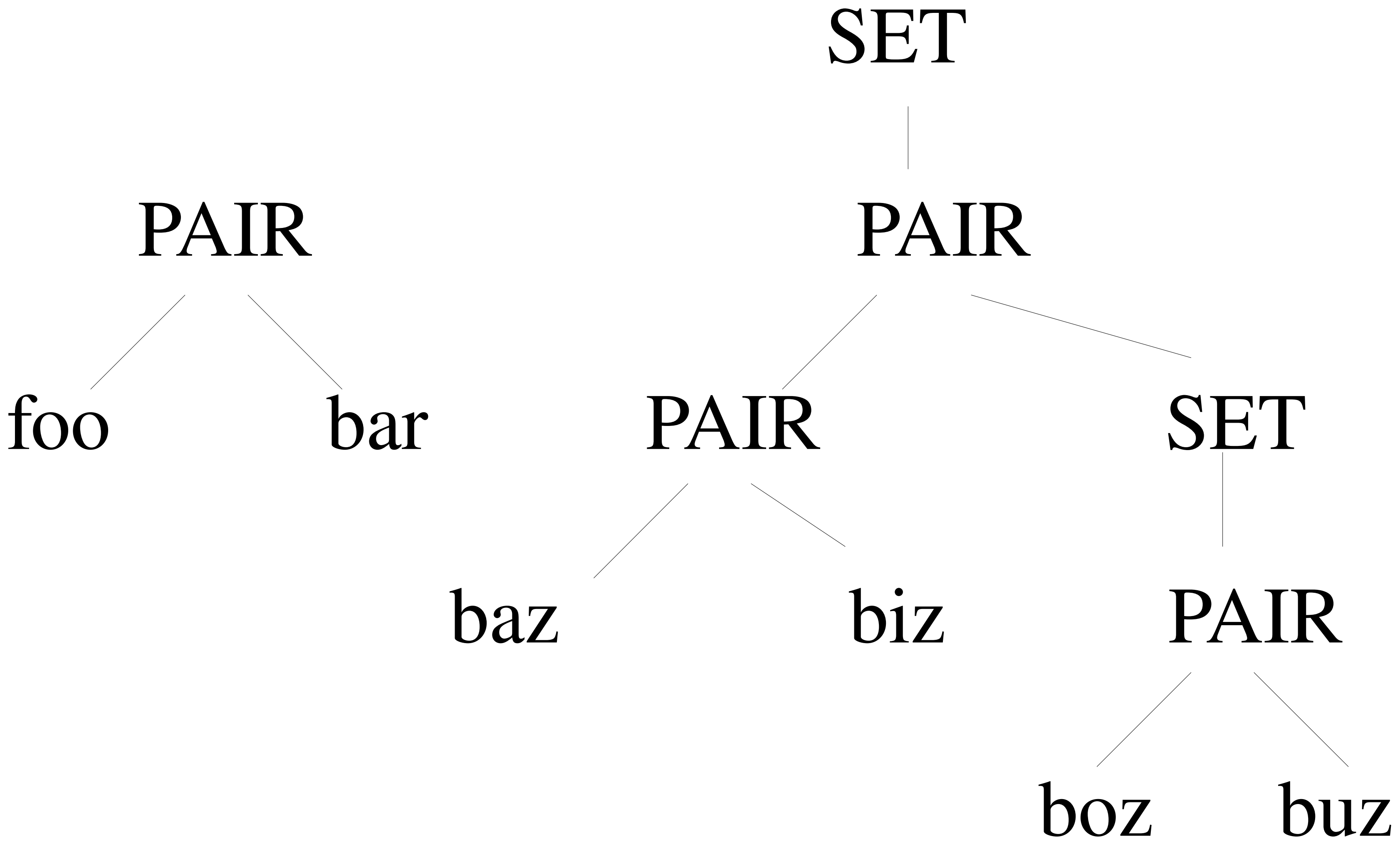

SET
PAIR
PAIR
SET
foo
bar
PAIR
PAIR
buz
PAIR
boz
baz
biz

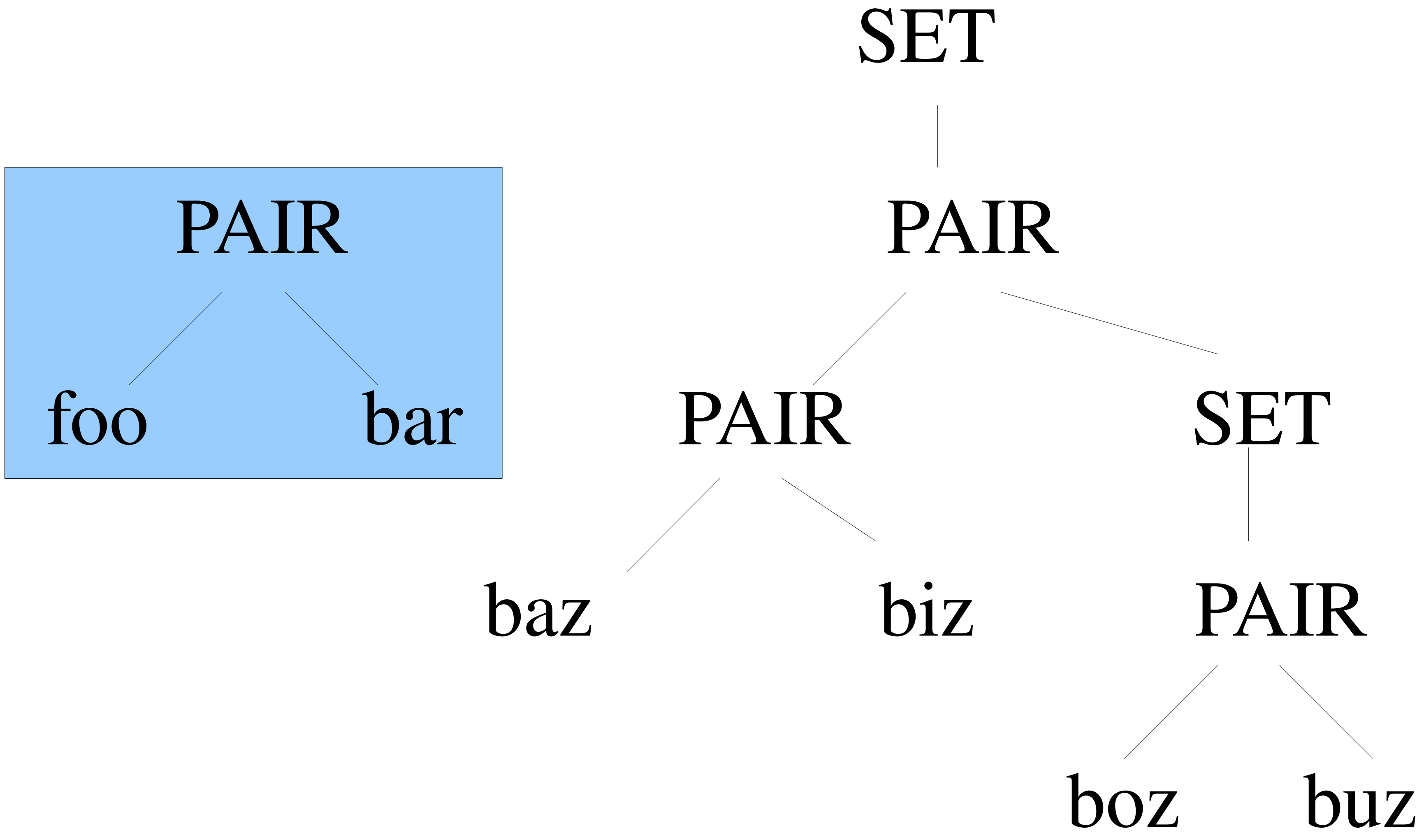
“ ~~foo bar PAIR~~ baz biz PAIR boz PAIR buz PAIR SET ~~PAIR SET~~”


SET
PAIR
PAIR
foo
bar
SET
PAIR
PAIR
buz
PAIR
boz
baz
biz

“ *a*” “ foo bar PAIR SET” “ *b*”
“ foo bar PAIR baz biz PAIR boz buz
      PAIR SET PAIR SET”
$\triangleleft$ _


SET
PAIR
foo
bar


SET
PAIR
PAIR
foo
bar
PAIR
SET
baz
biz
PAIR
boz
buz


PAIR
foo
bar
SET
PAIR
PAIR
SET
baz
biz
PAIR
boz
buz

“ ~~foo bar PAIR~~ baz biz PAIR boz PAIR buz
PAIR SET ~~PAIR SET~~”

“ baz biz PAIR boz PAIR buz PAIR   SET”

```
   1    2  3-2=1  2    3-2=1  2   3-2=1 2-1=1
```

“ ~~foo bar PAIR~~ baz biz PAIR boz buz
PAIR SET ~~PAIR SET~~”

“ baz biz PAIR boz buz PAIR   SET”

1   2   3-2=1 2   3   4-2=2 3-1=2

“ ~~foo bar PAIR~~ baz PAIR biz PAIR boz PAIR buz bez PAIR SET ~~PAIR SET~~”

“ baz PAIR biz PAIR boz PAIR buz bezPAIR SET”

1 2-2=0   1   2-2=0 1   2-2=0   1   2 3-2=1 2-1=1

Conclusions

1: It is simple to add type Strings to FORTH expressions.

Conclusions

2: Analysing these types in a second pass of compilation allows the types of the result to be determined and correctness of the types passed to be verified.

Conclusions

3: We have a simple method for verifying that a postfix String represents a well-formed single expression.