Backtracking with scanning

Consider this:

```
][ "scan this" ? every i := 1 to 10 do
                   write(tab(i));

s
c
a
n

t
h
i
s
Failure
```

And this:

```
][ "scan this" ? every write(tab(1 to 10));

s
sc
sca
scan
scan
scan t
scan th
scan thi
scan this
Failure
```

What's going on?

Backtracking with scanning, continued

In fact, `tab()` is a generator.

A simple approximation of `tab(n)`:

```
procedure Tab(n)
    oldpos := &pos
    &pos := n
    suspend &subject[oldpos:n]
    &pos := oldpos
end
```

Resumption of `tab` undoes any change to `&pos`.

`move(n)` is also a generator, changing `&pos`, suspending, and restoring the old value if resumed.

In essence, any `tab`'s and `move`'s in a failing expression are undone.

```
tab(upto(...)) & ="..." & move(...) &
s := tab(many(...)) & p1(...)
```

Backtracking with scanning, continued

Note the difference between bounded and unbounded `tab(...)` calls:

```
][ "abc 123" ? {
                tab(many(&letters))
                tab(many(&digits))
                snap()
               };
&subject =  a b c   1 2 3
&pos =  4        |

][ "abc 123" ? {
                tab(many(&letters)) &
                tab(many(&digits))
                snap()
               };
&subject =  a b c   1 2 3
&pos =  1  |
```

Two more cases:

```
][ "abc123" ? { tab(many(&letters)) &
               tab(many(&digits))
               snap() };
&subject =  a b c 1 2 3
&pos =  7              |

][ "123" ? { tab(many(&letters)) &
            tab(many(&digits))
            snap() };
&subject =  1 2 3
&pos =  1  |
```


Backtracking in scanning, continued

Here's a program that recognizes time duration specifications such as "10m" or "50s":

```
procedure main(args)
  while line := (writes("String? "),read()) do
    line ?
      if tab(many(&digits)) & move(1) == !"ms" &
                pos(0) then write("yes")
                       else write("no")
end
```

Interaction:

```
String? 10m
yes
String? 50s
yes
String? 100
no
String? 30x
no
```

Backtracking in scanning, continued

A revision that also recognizes specifications such as "10:43" or "7:18":

```
procedure main()
   while line := (writes("String? "), read()) do
       line ?
          if (Nsecs() | mmss()) & pos(0) then
              write("yes")
          else
              write("no")
end

procedure Nsecs()
    tab(many(&digits)) & move(1) == !"ms" &
    return
end

procedure mmss()
    mins := tab(many(&digits)) & =":" &
    nsecs := tab(many(&digits)) &
    *nsecs = 2 & return
end
```

Interaction:

```
String? 10m
yes
String? 9:41
yes
String? 8:100
no
String? 100x
no
```


Backtracking in scanning, continued

Imagine a program that looks for duration specifications and marks them:

```
% cat mark.1
The May 30 tests showed durations
between 75s and 2m.  Further analysis
revealed the span to be 1:14 to 2:03.
%
%
% mark < mark.1
The May 30 tests showed durations

between 75s and 2m.  Further analysis
        ^^^     ^^
revealed the span to be 1:14 to 2:03.
                        ^^^^    ^^^^
%
```

The code:

```
procedure main()
    while line := read() do {
       write(line)
       markline := repl(" ", *line)
       line ? while skip := tab(upto(&digits)) do {
         start := &pos
         ((Nsecs|mmss)() &
          len := &pos - start &
          markline[start+:len] := repl("^", len)) |
          tab(many(&digits))
          }
       write(markline)
       }
end
```

`Nsecs()` and `mmss()` are unchanged.

Backtracking in scanning, continued

Problem: Write a program that reads `Image()` output and removes the list labels.

Example:

```
% cat samples
r := L1:[1,2,3]  (list)
r := L1:[1,L2:[2],L3:[L4:[3,4]]]  (list)
r := L1:[L2:[],L2,L2,L2,L2]  (list)
%
% cleanlx < samples
r := [1,2,3]  (list)
r := [1,[2],[[3,4]]]  (list)
r := [[],L2,L2,L2,L2]  (list)
%
```