The creative extension principle

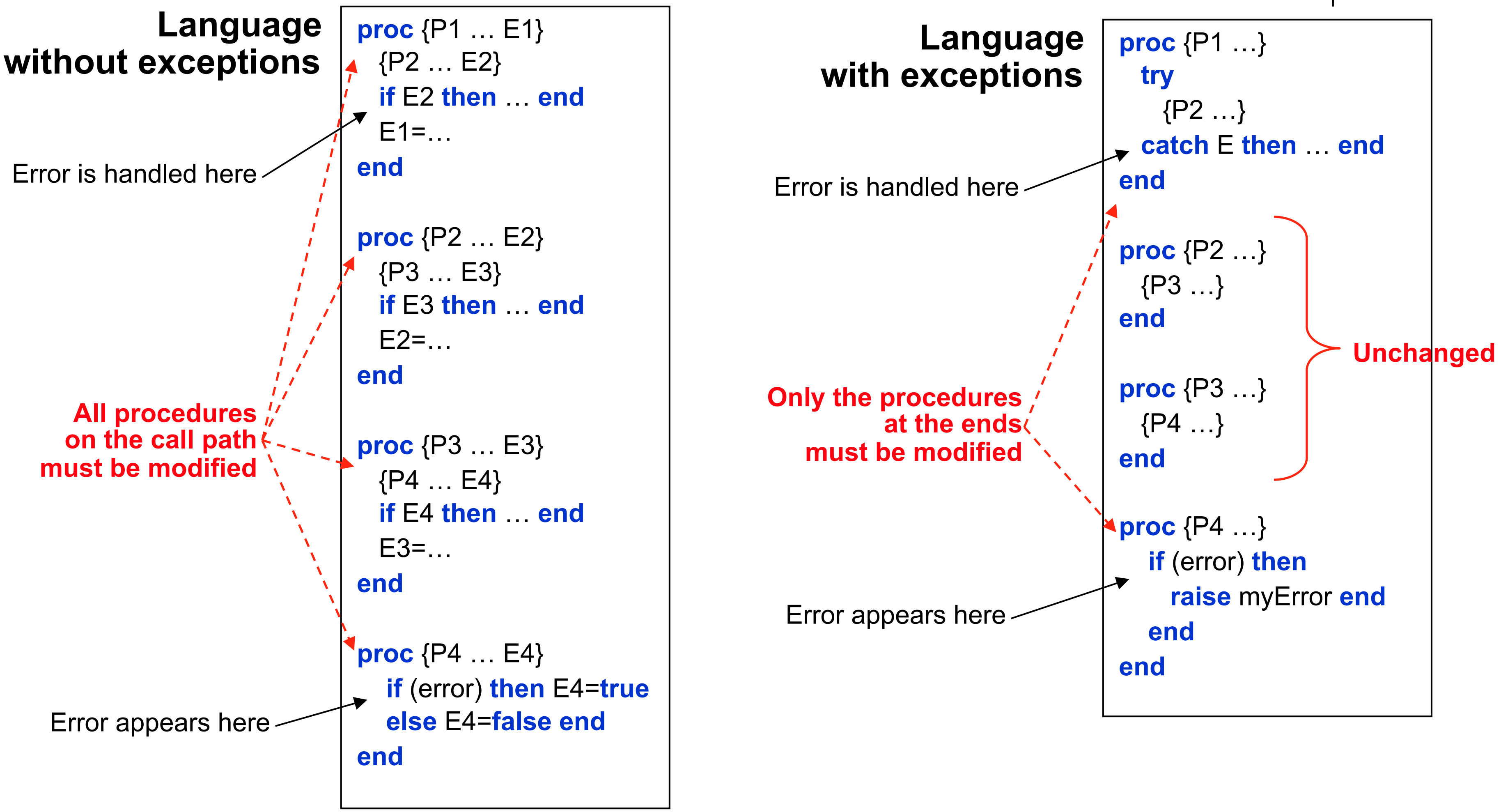
- How do we invent a new paradigm?

- When, in a given paradigm, programs start getting complicated for technical reasons that are unrelated to the problem being solved (e.g., nonlocal program transformations are needed), then there is a new programming concept waiting to be discovered!
  - Adding this concept to the paradigm lets programs get simple again

- We saw one example: the concept of concurrency
  - If your paradigm does not have concurrency, and you need it, then you are forced to implement it, making your programs complicated!

- Another example is the concept of exceptions
  - If the paradigm does not support them (e.g., like in C), then all routines on the call path must test and return error codes
  - If the paradigm supports them (e.g., like in Java), then only the ends of the call path need to be changed (raise and catch exceptions)

Exception handling



Language without exceptions

Error is handled here

All procedures on the call path must be modified

Error appears here

```
proc {P1 … E1}
   {P2 … E2}
   if E2 then … end
   E1=…
end

proc {P2 … E2}
   {P3 … E3}
   if E3 then … end
   E2=…
end

proc {P3 … E3}
   {P4 … E4}
   if E4 then … end
   E3=…
end

proc {P4 … E4}
    if (error) then E4=true
    else E4=false end
end
```

Language with exceptions

Error is handled here

Only the procedures at the ends must be modified

Error appears here

Unchanged

```
proc {P1 …}
   try
      {P2 …}
   catch E then … end
end

proc {P2 …}
   {P3 …}
end

proc {P3 …}
   {P4 …}
end

proc {P4 …}
    if (error) then
       raise myError end
    end
end
```

Complete set of concepts (so far)

<s> ::=		
skip	*Empty statement*	Descriptive declarative
<x>$_1$=<x>$_2$	*Variable binding*	
<x>=<record> | <number> | <procedure>	*Value creation*	
<s>$_1$ <s>$_2$	*Sequential composition*	
local <x> **in** <s> **end**	*Variable creation*	
if <x> **then** <s>$_1$ **else** <s>$_2$ **end**	*Conditional*	Declarative
case <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**	*Pattern matching*	
{<x> <x>$_1$ … <x>$_n$}	*Procedure invocation*	
thread <s> **end**	*Thread creation*	
{WaitNeeded <x>}	*By-need synchronization*	
{NewName <x>}	*Name creation*	↓ Less declarative
<x>$_1$= !!<x>$_2$	*Read-only view*	
try <s>$_1$ **catch** <x> **then** <s>$_2$ **end**	*Exception context*	
raise <x> **end**	*Raise exception*	
{NewPort <x>$_1$ <x>$_2$}	*Port creation*	
{Send <x>$_1$ <x>$_2$}	*Port send*	
<space>	*Encapsulated search*	

- These concepts and their paradigms are all explained in the CTM book

Complete set of concepts (so far)

<s> ::=

skip	*Empty statement*	
<x>$_1$=<x>$_2$	*Variable binding*	
<x>=<record> | <number> | <procedure>	*Value creation*	
<s>$_1$ <s>$_2$	*Sequential composition*	
local <x> **in** <s> **end**	*Variable creation*	
if <x> **then** <s>$_1$ **else** <s>$_2$ **end**	*Conditional*	
case <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**	*Pattern matching*	
{<x> <x>$_1$ … <x>$_n$}	*Procedure invocation*	
thread <s> **end**	*Thread creation*	
{WaitNeeded <x>}	*By-need synchronization*	
{NewName <x>}	*Name creation*	
<x>$_1$= !!<x>$_2$	*Read-only view*	
try <s>$_1$ **catch** <x> **then** <s>$_2$ **end**	*Exception context*	
raise <x> **end**	*Raise exception*	
{NewCell <x>$_1$ <x>$_2$}	***Cell creation***	**Alternative**
{Exchange <x>$_1$ <x>$_2$ <x>$_3$}	***Cell exchange***	
<space>	*Encapsulated search*	