EXCEPTIONS

- error to handle run-time errors: deterministically aborts program execution
- for many applications, this is not the appropriate behaviour
- exceptions provide a more fine-grained method to respond to run-time errors

Slide 1  Concrete Syntax:
- Catching an exception:
  ```
  try e1 catch x => e2
  ```
- Raising an exception:
  ```
  raise::τ e
  ```

Slide 2  Evaluation:
1. evaluate e1
2. If raise(v) is raised during the execution of e1, bind x to v and evaluate e2

Abstract Syntax:
- try(e1, x:e2)
- raise(τ, e)

Example:
```csharp
try if (y < 0) then raise::Int -1 else x / y catch res => if res == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

EXCEPTIONS

- error to handle run-time errors: deterministically aborts program execution
- for many applications, this is not the appropriate behaviour
- exceptions provide a more fine-grained method to respond to run-time errors

Slide 1  Concrete Syntax:
- Catching an exception:
  ```
  try e1 catch x => e2
  ```
- Raising an exception:
  ```
  raise::τ e
  ```

Slide 2  Evaluation:
1. evaluate e1
2. If raise(v) is raised during the execution of e1, bind x to v and evaluate e2

Abstract Syntax:
- try(e1, x:e2)
- raise(τ, e)

Example:
```csharp
try if (y < 0) then raise::Int -1 else x / y catch res => if res == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```

Example:
```csharp
try e1 catch e2 => if e2 == -1 ....
```
Evaluation

Two possibilities to evaluate exceptions:
1. unwind the stack to search for top-most catch-frame
2. directly jump to catch-frame

**Slide 5**

First approach: We introduce a new machine state:

\[ s \leftarrow \text{raise}(v) \]

Propagate exception upwards in the control stack \( s \) looking for a handler

Evaluation rules:

\[
\begin{align*}
& (h, k) \rightarrow \text{try}(e_1, x, e_2) \quad \rightarrow_c (\text{catch}(k, x, e_2) \triangleright h, \text{try}(\square) \triangleright e_1) \\
& (\text{catch}(k, x, e_2) \triangleright h, \text{try}(\square) \triangleright e_1) \rightarrow_c (h, k) \triangleright v_1 \\
& (h, k) \rightarrow \text{raise}(\tau, e) \quad \rightarrow_c (h, \text{raise}(\tau, \square) \triangleright k) \triangleright e \\
& (h, \text{raise}(\tau, \square) \triangleright k) \triangleright e \rightarrow_c (h, k) \triangleright \text{raise}(v) \\
& \text{catch}(k', x, e_2) \triangleright h, k \triangleright \text{raise}(v) \rightarrow_c (h, k') \triangleright (v/x)e_2
\end{align*}
\]

The rules for all other expressions do not change (apart from the fact that the handler stack is present)

**Slide 6**

Second approach:

How can we jump directly to the appropriate handler?

- use an extra handler stack \( h \)
- a handler frame has the form \( \text{catch}(k, x, e) \) where
  - \( k \) is a stack
  - \( x, e \) a handler expression

**Slide 7**