WebAssembly Exception Handling (Phase 2)

Heejin Ahn

Updates on the Spec

try-catch-catch_all

- No `exnref / br_on_exn` (removed in 09/2020)
- `catch` extracts values onto the stack
- `catch_all` does not extract anything

```
try blocktype
  instruction*
catch i
  instruction*
catch j
  instruction*
…
catch_all
  instruction*
end
```

catch-less try

- We have removed the restriction that there should be at least one `catch` or `catch_all`
- In this case, the `try-end` does not catch any exceptions

```
try blocktype
  instruction*
end
```

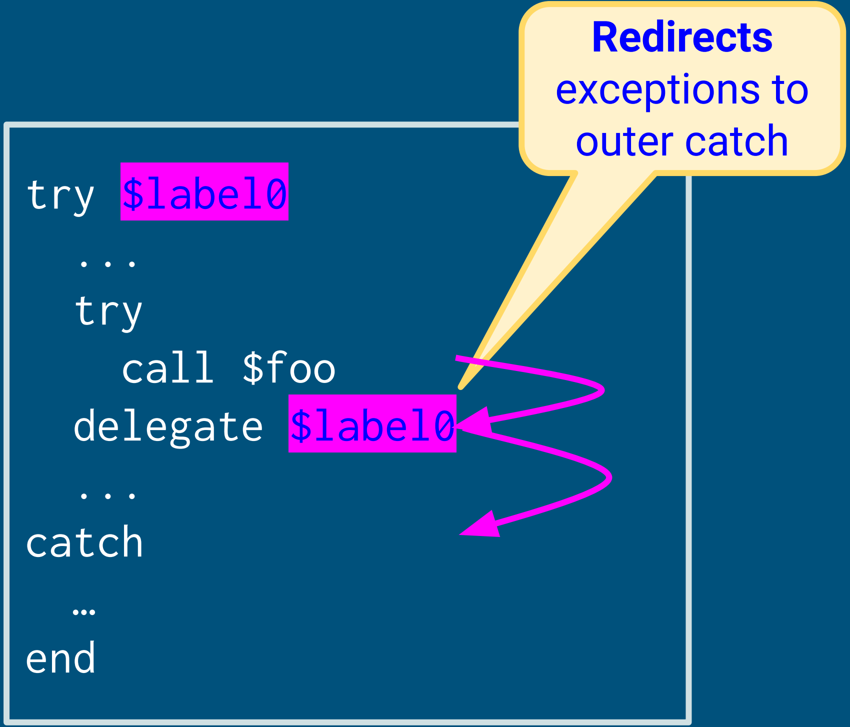
rethrow

- Gained an immediate argument, specifying which exception to rethrow
- Takes a `try` label (= immediate) and rethrows the exception caught by its corresponding `catch`

```
try $label0
  ...
catch  ;; $label0
  try $label1
    ...
  catch  ;; $label1
    rethrow $label0
    rethrow $label1
  end
end
```

delegate (previously catch_br)

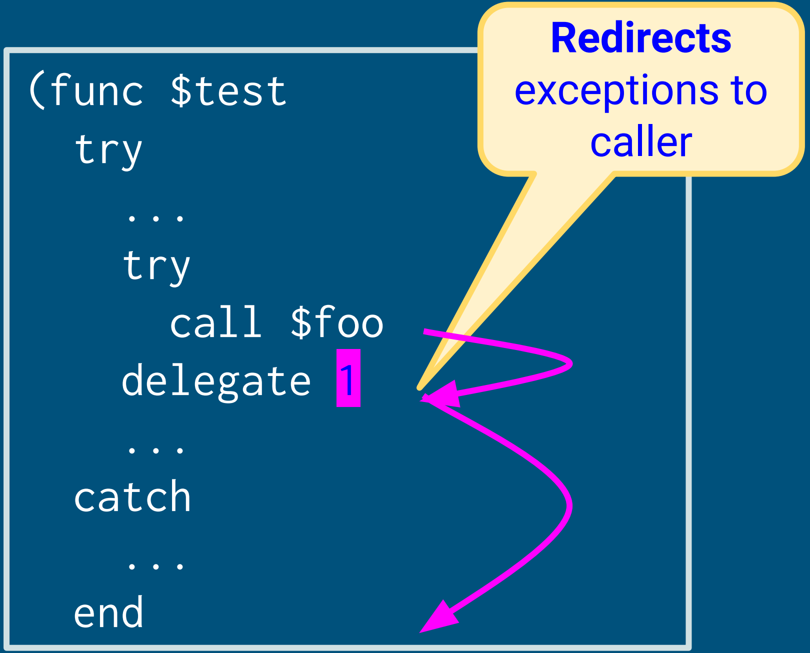
- Redirects exceptions to another catch in an outer scope
- Should target a `try` label



```
try $label0
  ...
  try
    call $foo
  delegate $label0
  ...
catch
  …
end
```

delegate

- When the function scope is targeted, it will be considered like a catchless `try`
- Redirects exception handling to the caller



```
(func $test
  try
    ...
    try
      call $foo
    delegate 1
    ...
  catch
    ...
  end
```

No unwind

- We have removed `unwind` because it does not have a meaningful difference with `catch_all` in the current spec
- It can be added in the potential future two-phase EH proposal

Tag Section

- We renamed events to tags, and event section to tag section
- Can be used for other constructs later

```
(module
  …
  (tag $cpp_exn (params i32))
  …
  (func $test
    try
      …
    catch $cpp_exn
      …
    end
  )
)
```

JS API

```
interface Tag { // Typed tag in tag section
  constructor(TagType type);
  TagType type();
};

interface Exception { // Runtime exception thrown and caught
  constructor(Tag tag, sequence<any> payload);
  any getArg(Tag tag, unsigned long index);
  boolean is(Tag tag);
};
```

JS API Example

```
let tag = new WebAssembly.Tag({parameters: ["i32", "f32"]});
let exn = new WebAssembly.Exception(tag, [3, 3.5]);
WebAssembly.throw(exn);

exn.getArg(tag, 0) == 3;
exn.is(tag) == true;
tag.type() == {parameters: ["i32", "f32"]};
```

Phase 3?

Phase 3 Entry Requirements

- Test suite has been updated
- Test suite runs against the reference interpreter
  - Parsing works in the interpreter
  - Execution / validation work in the interpreter (will land in a few days)

Bonus: We have already met some of Phase 4 requirements

- Two Web VMs (V8 and FireFox) have implemented the proposal
- One toolchain (Emscripten) has implemented the proposal
  - We are working w/ Adobe on the Origin Trial
  - Stabilization and optimization work is in progress
- We have a draft formal spec PR

Poll for Phase 3