Countable nondeterminism

Countable nondeterminism

Motivation: if we can build a *fair* task scheduler, then we can use it to construct a countable nondeterminism machine.

Countable nondeterminism

Motivation: if we can build a *fair* task scheduler, then we can use it to construct a countable nondeterminism machine.

But [Dijkstra, ??], countable nondeterminism is *impossible*!

Countable nondeterminism

Motivation: if we can build a *fair* task scheduler, then we can use it to construct a countable nondeterminism machine.

But [Dijkstra, ??], countable nondeterminism is *impossible*!
[Under a continuity assumption]

Countable nondeterminism

Motivation: if we can build a *fair* task scheduler, then we can use it to construct a countable nondeterminism machine.

But [Dijkstra, ??], countable nondeterminism is *impossible*!
[Under a continuity assumption]

Failure of continuity / algebraicity. E.g.:

$$
\begin{aligned}
<_m, <_\infty &: \mathtt{nat} \to \mathtt{nat} \\
<_m n &= \text{If } n < m \text{ then } 0 \text{ else } \Omega \\
<_\infty n &= 0\,,
\end{aligned}
$$

but

$$
\begin{aligned}
<_m ? &\equiv 0 \text{ or } \Omega \\
<_\infty ? &\equiv 0\,.
\end{aligned}
$$

Must-convergence

Nondeterministic languages differ from deterministic ones in that a given term M may admit multiple different small-step reduction rules.

If $M : o$ is a term of a nondeterministic language, we say that M *must converge*, and write $M \Downarrow^{\mathsf{must}}$, if every possible small-step evaluation path of M eventually terminates at some observable value.

Must-convergence

Nondeterministic languages differ from deterministic ones in that a given term M may admit multiple different small-step reduction rules.

If $M : o$ is a term of a nondeterministic language, we say that M *must converge*, and write $M \Downarrow^{\text{must}}$, if every possible small-step evaluation path of M eventually terminates at some observable value.

```
λ c . {
  for i = 0 to ? {
    c;
  }
}
```

Must-convergence

Nondeterministic languages differ from deterministic ones in that a given term M may admit multiple different small-step reduction rules.

If $M\colon o$ is a term of a nondeterministic language, we say that M *must converge*, and write $M \Downarrow^{\mathrm{must}}$, if every possible small-step evaluation path of M eventually terminates at some observable value.

```
λ c . {
  for i = 0 to ? {
    c;
  }
}
```

```
λ c . {
  while (TRUE) {
    c;
  }
}
```

Must-convergence, continued