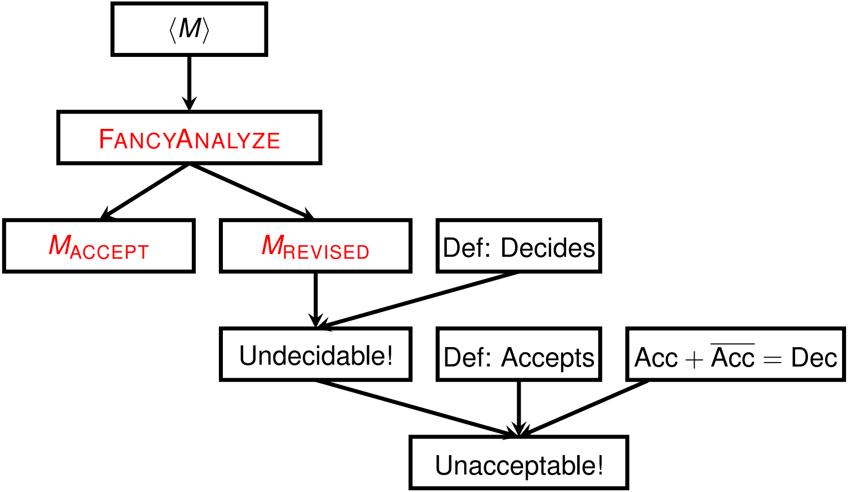
# Decidability & Acceptability

Joshua Eckroth

joshuaeckroth@gmail.com

# The Plan

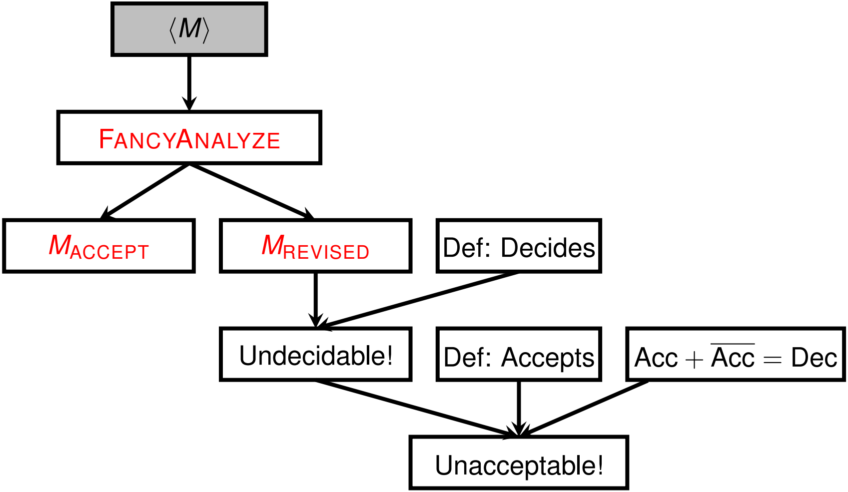

$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# String encodings of Turing Machines

First, we must allow a Turing Machine $M$ to be encoded as a string $\langle M \rangle$.

Once encoded, it can be the *input* of a Turing Machine.

# The Plan



$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

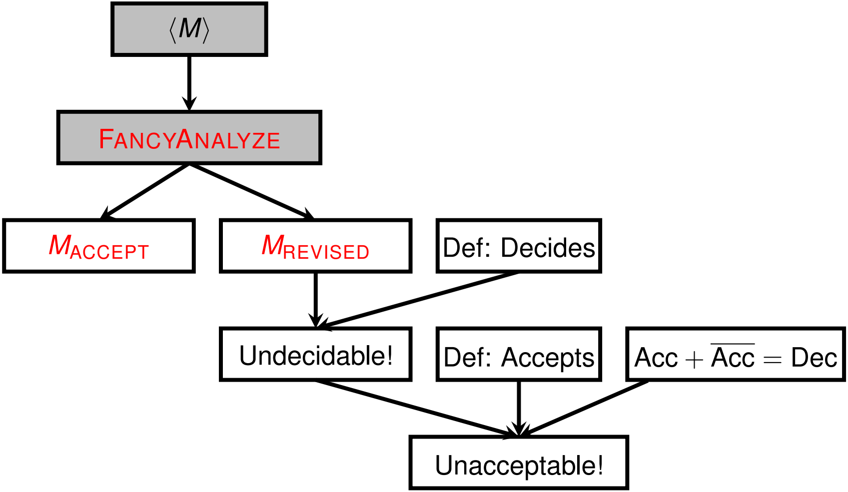
# A fancy analyzer of Turing Machines

Suppose we have a function, FANCYANALYZE($\langle M \rangle$,$w$), that can do the following:

- Analyze how $M$ (described by the string $\langle M \rangle$) behaves on input $w$. (Don't actually simulate $M$!)
- If $M$ *accepts* $w$, then FANCYANALYZE returns *accept*.
- If $M$ *rejects* $w$, then FANCYANALYZE returns *reject*.
- If $M$ can be shown to get stuck in a loop on $w$, and therefore never accepts or rejects, then FANCYANALYZE returns *loops*.

Could a fancy programmer write FANCYANALYZE . . . ?

# The Plan



$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# Using FANCYANALYZE to create a self-decider

$M_{\text{ACCEPT}}$ identifies (decides) the class of “machines that accept their own description.”

**function** $M_{\text{ACCEPT}}(\langle M \rangle)$
    Answer $\leftarrow$ FANCYANALYZE($\langle M \rangle$, $\langle M \rangle$)
    **if** Answer $=$ *accept* **then**
        **return** *accept*
    **else if** Answer $=$ *reject* or Answer $=$ *loops* **then**
        **return** *reject*
    **end if**
**end function**

# A quick self-check

What does $M_{\text{ACCEPT}}$ do on input $\langle M_{\text{ACCEPT}} \rangle$?

$M_{\text{ACCEPT}}(\langle M_{\text{ACCEPT}} \rangle) =$

- If $\text{FANCYANALYZE}(\langle M_{\text{ACCEPT}} \rangle, \langle M_{\text{ACCEPT}} \rangle)$ returns *accept*, then $M_{\text{ACCEPT}}(\langle M_{\text{ACCEPT}} \rangle)$ returns *accept*.
- If $\text{FANCYANALYZE}(\langle M_{\text{ACCEPT}} \rangle, \langle M_{\text{ACCEPT}} \rangle)$ returns *reject* or ~~*loop*~~, then $M_{\text{ACCEPT}}(\langle M_{\text{ACCEPT}} \rangle)$ returns *reject*.

Note: $\text{FANCYANALYZE}(\langle M_{\text{ACCEPT}} \rangle, \langle M_{\text{ACCEPT}} \rangle)$ won't return *loop* because $M_{\text{ACCEPT}}$ is a decider.

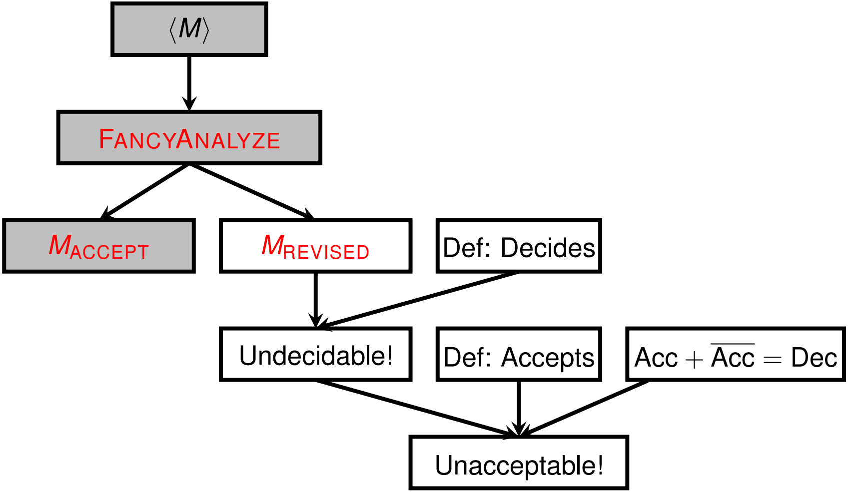
# A quick self-check (in other words)

What does $M_{\text{ACCEPT}}$ do on input $\langle M_{\text{ACCEPT}} \rangle$?

$M_{\text{ACCEPT}}(\langle M_{\text{ACCEPT}} \rangle) =$

- If $M_{\text{ACCEPT}}$ accepts its own description, then it accepts its own description (duh).
- If $M_{\text{ACCEPT}}$ rejects its own description, then it rejects its own description (duh).

# The Plan



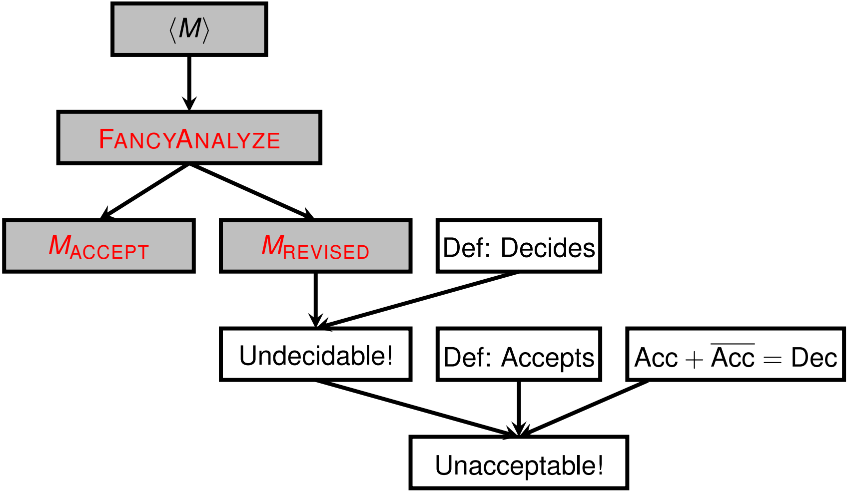
$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# A backwards self-decider (why not?)

**function** $M_{\text{REVISED}}(\langle M \rangle)$
  Answer $\leftarrow$ $\textsc{FancyAnalyze}(\langle M \rangle, \langle M \rangle)$
  **if** Answer $=$ *accept* **then**
    **return** *reject* $\triangleright$ opposite!
  **else if** Answer $=$ *reject* or Answer $=$ *loops* **then**
    **return** *accept* $\triangleright$ opposite!
  **end if**
**end function**

If you believe that we could implement $M_{\text{ACCEPT}}$, then you have to allow us to implement $M_{\text{REVISED}}$.

# The Plan



⟨M⟩
FancyAnalyze
M_Accept
M_Revised
Def: Decides
Undecidable!
Def: Accepts
Acc + $\overline{\text{Acc}}$ = Dec
Unacceptable!

# The sucker punch

What does $M_{\text{REVISED}}$ do on input $\langle M_{\text{REVISED}} \rangle$?

$M_{\text{REVISED}}(\langle M_{\text{REVISED}} \rangle) =$

- If $\textsc{FancyAnalyze}(\langle M_{\text{REVISED}} \rangle, \langle M_{\text{REVISED}} \rangle)$ returns *accept*, then $M_{\text{REVISED}}(\langle M_{\text{REVISED}} \rangle)$ returns *reject*!
- If $\textsc{FancyAnalyze}(\langle M_{\text{REVISED}} \rangle, \langle M_{\text{REVISED}} \rangle)$ returns *reject*, then $M_{\text{REVISED}}(\langle M_{\text{REVISED}} \rangle)$ returns *accept*!

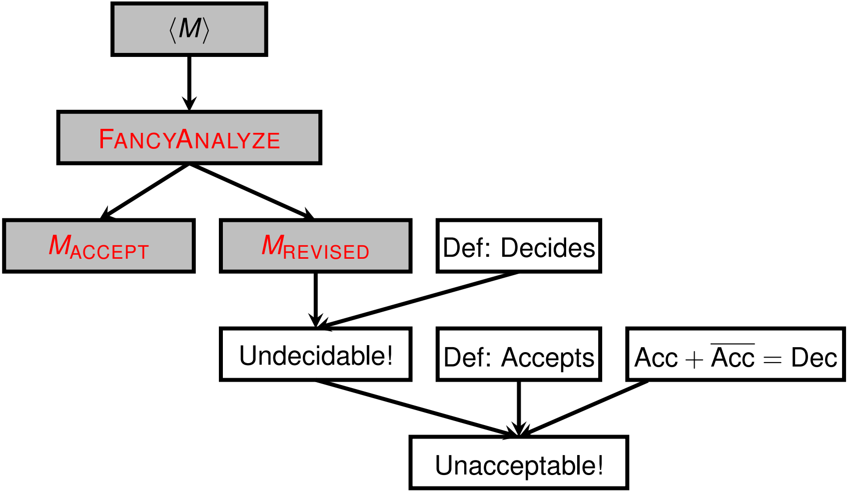
# The sucker punch (in other words)

What does $M_{\text{REVISED}}$ do on input $\langle M_{\text{REVISED}} \rangle$?

$M_{\text{REVISED}}(\langle M_{\text{REVISED}} \rangle) =$

- If $M_{\text{REVISED}}$ accepts its own description, then it rejects its own description!
- If $M_{\text{REVISED}}$ rejects its own description, then it accepts its own description!

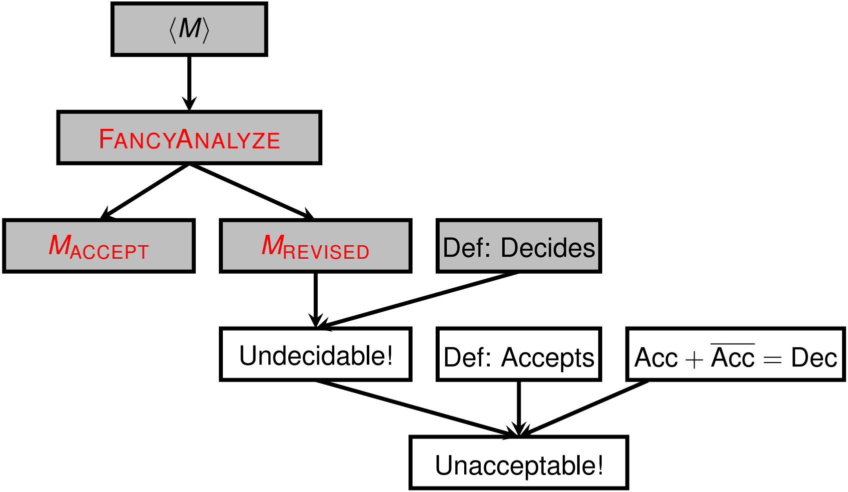
# What went wrong?



# Definition: Decides

We say that a machine $M$ *decides* a language $L$ if for every $w \in L$, $M$ with input $w$ halts with the answer *accept* and for every $\overline{w} \notin L$, $M$ with input $\overline{w}$ halts with the answer *reject*.

# The Plan



$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# Undecidable!

So FANCYANALYZE cannot exist (as a decider). The *language* it would have decided is,

$$\mathrm{ACCEPT}_{\mathrm{TM}} = \{(\langle M \rangle, w) : M \text{ accepts } w\}.$$

However, $\mathrm{ACCEPT}_{\mathrm{TM}}$ is easy to *accept*. Just literally simulate $M$ on $w$ (if $M$ accepts $w$, so does the simulator).

# The Halting Problem

FANCYANALYZE($\langle M \rangle$, $w$) could have been written differently:

- If $M$ halts on $w$ (accepts or rejects $w$), then FANCYANALYZE halts.
- Otherwise, loop forever.

# The Halting Problem

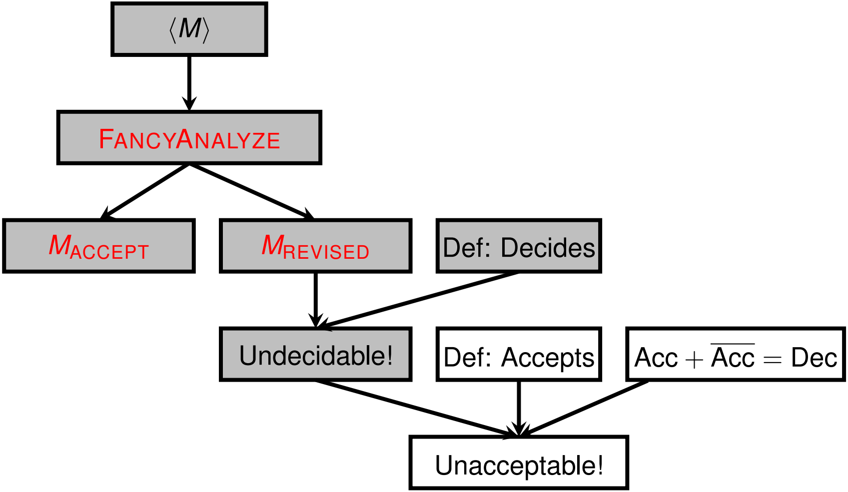
Then we make a variant, WACKOANALYZE, that does the opposite (loops forever if *M* halts, and halts if *M* loops forever).

Now, WACKOANALYZE, when executed on its own description, halts when it loops forever and loops forever when it halts (!).

# Undecidable!

Either way, we have an undecidable problem (i.e., undecidable language).

# The Plan

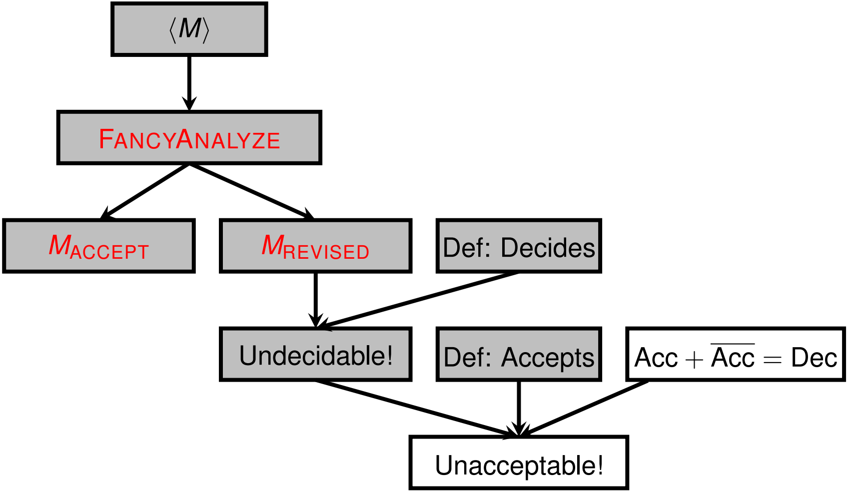


$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# Definition: Accepts

We say that a machine $M$ *accepts* (or *recognizes*) a language $L$ if for every $w \in L$, $M$ with input $w$ halts with the answer *accept*.

# The Plan



$\langle M \rangle$
FANCYANALYZE
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

## $\mathrm{Acc} + \overline{\mathrm{Acc}} = \mathrm{Dec}$

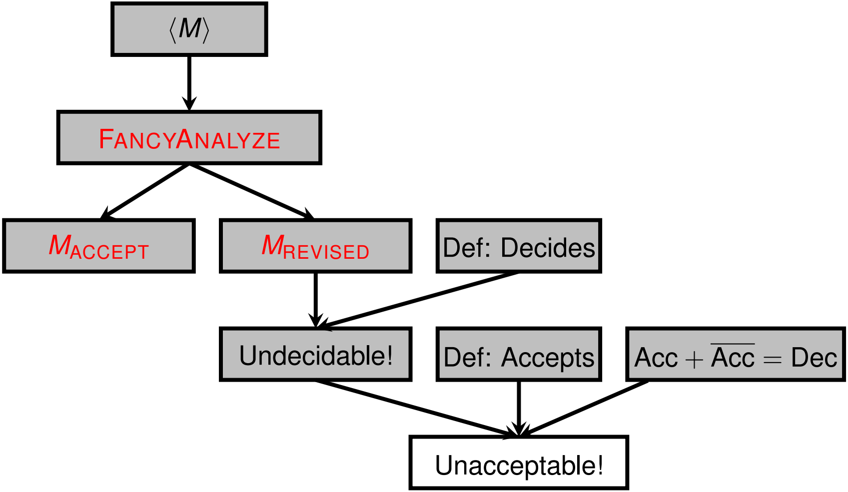
- Suppose we have a machine $M$ that accepts a language Acc.
- Consider the language $\overline{\mathrm{Acc}}$, i.e., the language $\{\overline{w} : \overline{w} \notin \mathrm{Acc}\}$.
- Suppose we have a machine $\overline{M}$ that accepts $\overline{\mathrm{Acc}}$.

## $\text{Acc} + \overline{\text{Acc}} = \text{Dec}$

- By simulating $M$ and $\overline{M}$ in *parallel*, we can build a decider for both Acc and $\overline{\text{Acc}}$. Here it is for Acc:

**function** $M_{\textsc{Decider for Acc}}(w)$
    **while** true **do**
        $m \leftarrow \textsc{SimulateOneStep}(\langle M \rangle, w)$
        $\overline{m} \leftarrow \textsc{SimulateOneStep}(\langle \overline{M} \rangle, w)$
        **if** $m = \textit{accept}$ **then return** *accept*
        **else if** $\overline{m} = \textit{accept}$ **then return** *reject*
        **end if**
    **end while**
**end function**

# The Plan



$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
$\text{Acc} + \overline{\text{Acc}} = \text{Dec}$
Unacceptable!

# Unacceptable!

We can also prove there exists a language that cannot be accepted (recognized).

It's simply,

$$\overline{\text{ACCEPT}_{\text{TM}}} = \{(\langle M \rangle, w) : M \text{ does not accept } w\}.$$

# Unacceptable!

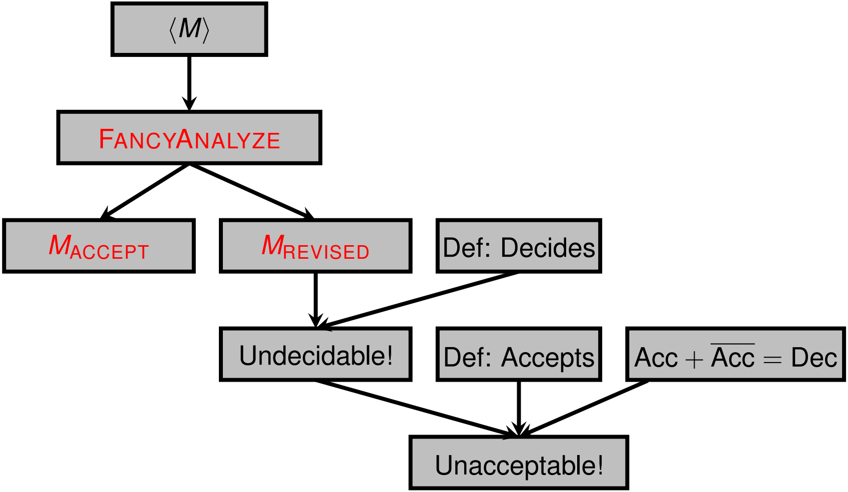
$\overline{\text{ACCEPT}_{\text{TM}}}$ is unacceptable.

Proof: Suppose it is acceptable. Then,

- $\overline{\text{ACCEPT}_{\text{TM}}}$ would be acceptable (by supposition).
- $\text{ACCEPT}_{\text{TM}}$ is acceptable (proved earlier).
- So, $\text{ACCEPT}_{\text{TM}}$ is decidable! Impossible (proved earlier).

So $\overline{\text{ACCEPT}_{\text{TM}}}$ is not acceptable.

# The Plan



$\langle M \rangle$
FancyAnalyze
$M_{\text{ACCEPT}}$
$M_{\text{REVISED}}$
Def: Decides
Undecidable!
Def: Accepts
Acc + $\overline{\text{Acc}}$ = Dec
Unacceptable!

# Summary

So we have a language that is acceptable but not decidable:

$$\mathsf{ACCEPT}_{\mathsf{TM}} = \{(\langle M \rangle, w) : M \text{ accepts } w\}.$$

And we have a language that is not even acceptable:

$$\overline{\mathsf{ACCEPT}_{\mathsf{TM}}} = \{(\langle M \rangle, w) : M \text{ does not accept } w\}.$$

# Practical implications

- You cannot write a *single* program that can check *every* program (including itself) and determine whether it is bug-free.
- Rice's theorem: There is no mechanical procedure to decide the answer to *any* non-trivial question about *every* machine.
- More importantly: There is no mechanical procedure to decide the answer to every mathematical question about the natural numbers.