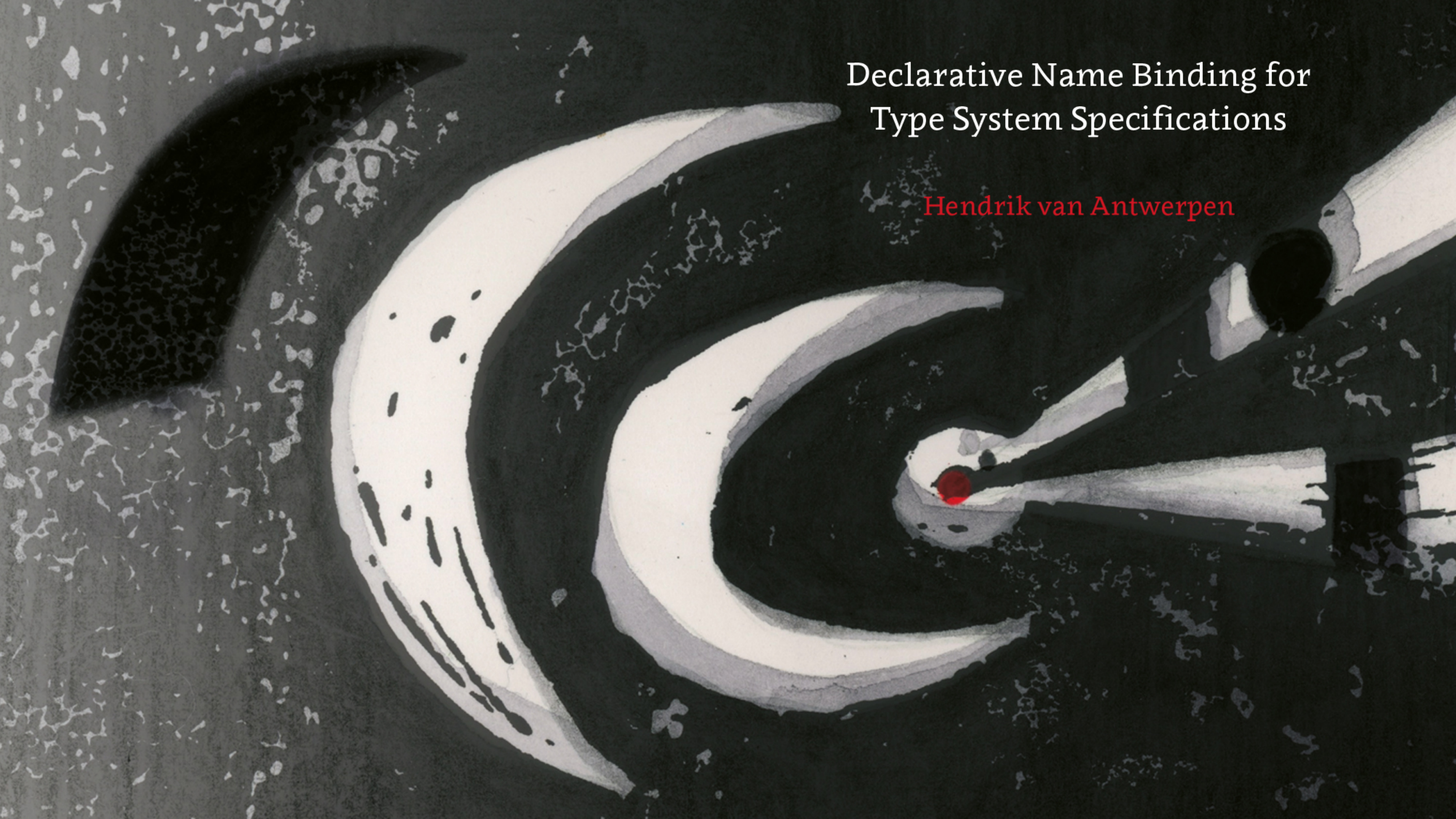


Declarative Name Binding for Type System Specifications

Hendrik van Antwerpen

High-level description (what, not how) of which references resolve to which declarations

Declarative Name Binding for Type System Specifications

in properties we can determine from reading a program without executing it, using logic and formal rules.

A simple example

Syntax

$$
\begin{array}{rlll}
\text{integers} & z \in \mathbb{Z} & := & \{\ldots,\texttt{-1},\texttt{0},\texttt{1},\ldots\} \\
\text{identifiers} & \texttt{x} \in \mathit{Id} & := & \text{some countable set} \\
\text{expressions} & \texttt{e} \in \mathit{Expr} & ::= & z \mid \texttt{e + e} \mid \texttt{fun(x : t)\{ e \}} \mid \texttt{x} \mid \texttt{e e} \mid \texttt{let x = e in e} \\
\text{types} & \texttt{t} \in \mathit{Type} & ::= & \texttt{num} \mid \texttt{t -> t}
\end{array}
$$

Resolution Syntax Parameters

$$
\begin{array}{rlll}
\text{labels} & l \in \mathcal{L} & ::= & \texttt{P} \\
\text{relations} & r \in \mathcal{R} & ::= & \texttt{:} \\
\text{data terms} & d \in \mathcal{D} & ::= & \texttt{x}_i : \texttt{t}
\end{array}
$$

Matching Declarations, Label Order and Data Order

$\boxed{d \in \textsc{decl}(\texttt{x}_i)}$ $\boxed{\hat{l} <_l \hat{l}}$ $\boxed{t \leq_\top t}$

$$\frac{\texttt{x}_i \simeq \texttt{x}_j}{(\texttt{x}_j : \texttt{t}) \in \textsc{decl}(\texttt{x}_i)} \qquad \$ <_l P \qquad t_1 \leq_\top t_2$$

Typing Rules

$\boxed{\mathcal{G}, s \vdash \texttt{e} : \texttt{t}}$

$$\text{(STLC-Num)}\ \frac{}{s \vdash z : \texttt{num}} \qquad \text{(STLC-Plus)}\ \frac{s \vdash \texttt{e}_1 : \texttt{num} \quad s \vdash \texttt{e}_2 : \texttt{num}}{s \vdash \texttt{e}_1\ \texttt{+}\ \texttt{e}_2 : \texttt{num}}$$

$$\text{(STLC-Fun)}\ \frac{\nabla s_2 \quad s_2 \xrightarrow{\texttt{P}} s_1 \quad s_2 \xrightarrow{\texttt{:}} \texttt{x}_i : \texttt{t}_1 \quad s_2 \vdash \texttt{e} : \texttt{t}_2}{s_1 \vdash \texttt{fun(x}_i\ \texttt{:}\ \texttt{t}_1\texttt{)\{ e \}} : \texttt{t}_1\ \texttt{->}\ \texttt{t}_2}$$

$$\text{(STLC-Id)}\ \frac{\textsc{decl}(\texttt{x}_i), \texttt{P}^*, \leq_\top, <_l \vdash p : s \overset{\texttt{:}}{\mapsto} \texttt{x}_j : \texttt{t}}{s \vdash \texttt{x}_i : \texttt{t}} \qquad \text{(STLC-App)}\ \frac{s \vdash \texttt{e}_1 : \texttt{t}_1\ \texttt{->}\ \texttt{t}_2 \quad s \vdash \texttt{e}_2 : \texttt{t}_1}{s \vdash \texttt{e}_1\ \texttt{e}_2 : \texttt{t}_2}$$

$$\text{(STLC-Let)}\ \frac{s_1 \vdash \texttt{e}_1 : \texttt{t}_1 \quad \nabla s_2 \quad s_2 \xrightarrow{\texttt{P}} s_1 \quad s_2 \xrightarrow{\texttt{:}} \texttt{x}_i : \texttt{t}_1 \quad s_2 \vdash \texttt{e}_2 : \texttt{t}_2}{s_1 \vdash \texttt{let}\ \texttt{x}_i\ \texttt{=}\ \texttt{e}_1\ \texttt{in}\ \texttt{e}_2 : \texttt{t}_2}$$

Little Jamie in the kitchen

PB&J Tortilla

Ingredients

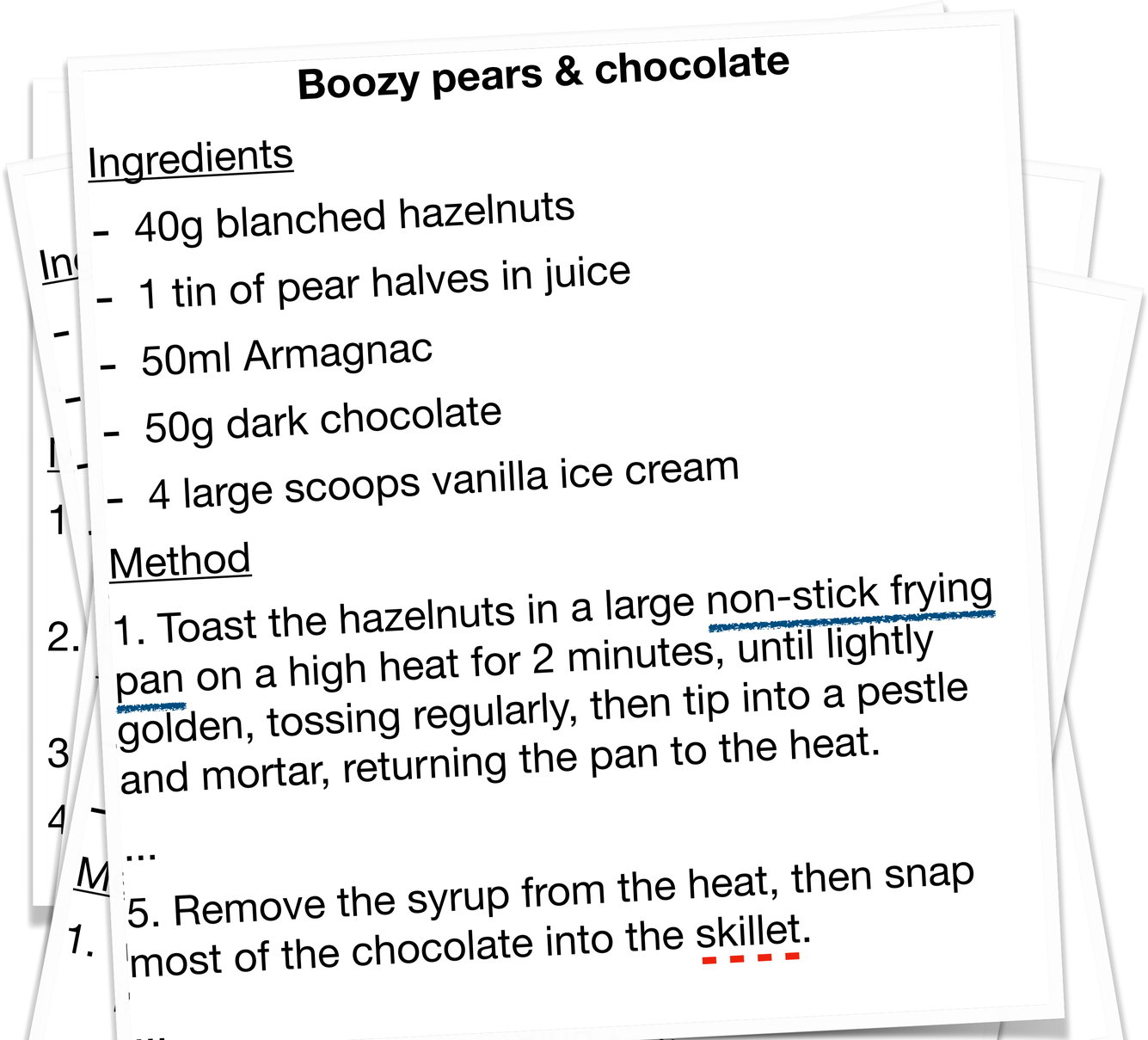
- 1 Tortilla
- A lot of peanut butter
- A lot of jelly

Method

1. Heat the tortilla above a stove burner until brown.
2. Put the peanut butter and the jelly in a bowl and mix.
3. Empty the bowl on top of the tortilla.
4. Crush the M&Ms with a rolling pin. Spread on top of the tortilla.

- Structure of a recipe
  - Output: what we get out
  - Inputs: what we need to put in
  - Steps: what needs to be done
  - Operations: things we can do
  - References: refer back to something
- Possible mistakes
  - Refer to missing ingredient or tool
  - Incorrect operation ("stir with a pot")
  - Find by just *reading* the recipe!

Jamie's first cookbook



Boozy pears & chocolate

Ingredients

- 40g blanched hazelnuts
- 1 tin of pear halves in juice
- 50ml Armagnac
- 50g dark chocolate
- 4 large scoops vanilla ice cream

Method

1. Toast the hazelnuts in a large non-stick frying pan on a high heat for 2 minutes, until lightly golden, tossing regularly, then tip into a pestle and mortar, returning the pan to the heat.

...

5. Remove the syrup from the heat, then snap most of the chocolate into the skillet.

...

RecipeChecker.exe

```
recipe.method.forEach((step, index) => {

    // ... other cases ...

    const re = /(\w+) the (\w+) into the (\w+)/g;
    for (const match in step.matchAll(re)) {
        const action = match[1];
        const food_item = match[2];
        const container = match[3];
        if (!allowedActions.includes(action)) {
            console.log(`step ${index+1}: unknown action ${action}`);
        }
        if (!findFoodItem(food_item, recipe.ingredients, recipe.method.slice(0, index))) {
            console.log(`step ${index+1}: unknown ingredient ${food_item}`);
        }
        if (!findContainer(container, recipe.method.slice(0, index))) {
            console.log(`step ${index+1}: unknown container ${container}`);
        }
    }

    // ... other cases ...

});

function findFoodItem(
    name: string,
    ingredients: string[],
    steps: string[],
): boolean {
    for (const step in steps) {
        if (step.includes(`to a ${name}`)) { return true; }
    }
    for (const ingredient in ingredients) {
        if (ingredient.includes(name)) { return true; }
    }
    return false;
}

function findContainer(
    name: string,
    steps: string[],
): boolean {
    for (const step in steps) {
        if (step.includes(`into a ${name}`)) { return true; }
    }
    return false;
}
```

✔ / ✗

Jamie Publishers

- More complex recipes
  - "Make a sauce using the recipe on page 23"
- Books in other languages
- Recipe checker becomes too complex
  - Hard to understand
  - Hard to maintain
  - Hard to improve

Generating recipe checker

```
Ingredients:
…
- (n) (u) (f)
…
-----------------------
(n) is-a number
(u) is-a unit
introduces food-item (f)
```

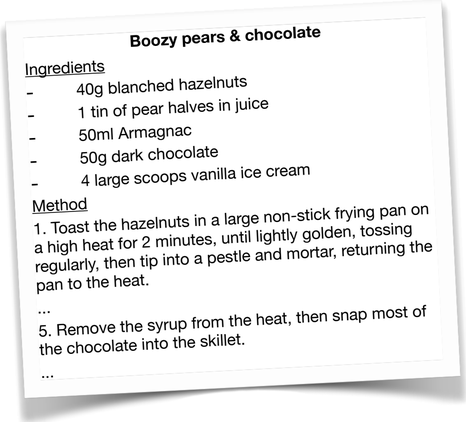
```
Method:
…
_. … put (f) into a (c) …
…
-----------------------
uses food-item (f)
introduces container (c)
```

```
Method:
…
_. … add (f) into the (c) …
…
-----------------------
uses food-item (f)
uses container (c)
```

Rules: match and effect. Together a specification.

Written in meta-language: talks about another language.

Rules work together, saying what but now how!



Boozy pears & chocolate

Ingredients

- 40g blanched hazelnuts
- 1 tin of pear halves in juice
- 50ml Armagnac
- 50g dark chocolate
- 4 large scoops vanilla ice cream

Method

1. Toast the hazelnuts in a large non-stick frying pan on a high heat for 2 minutes, until lightly golden, tossing regularly, then tip into a pestle and mortar, returning the pan to the heat.

...

5. Remove the syrup from the heat, then snap most of the chocolate into the skillet.

...

RecipeChecker.exe

✔ / ✗

High-level description (what, not how) of which references resolve to which declarations

Declarative Name Binding for Type System Specifications

in properties we can determine from reading a program without executing it, using logic and formal rules.

- Achieved by using a meta-language
- Rules are easier to understand and maintain than code
- We can automatically generating code for checkers