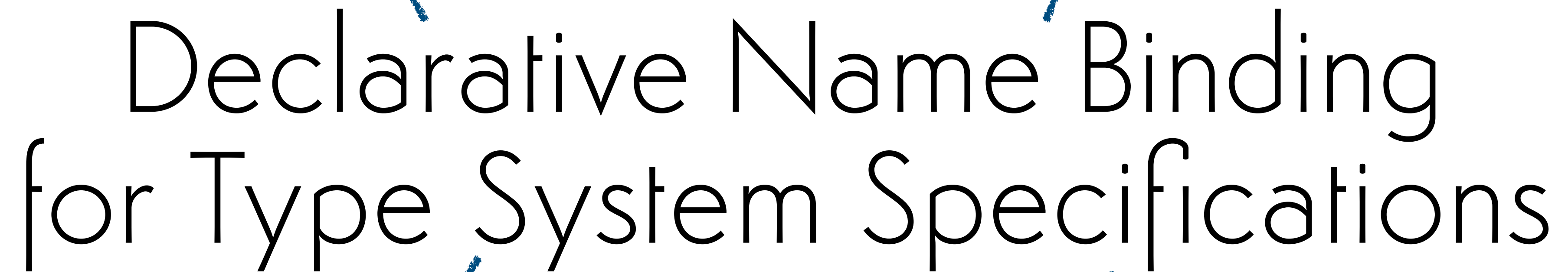
The background is a dark, textured surface. A prominent white spiral shape, resembling a stylized 'C' or a shell, curves from the bottom left towards the center. A small, bright red dot is located near the center of the spiral. To the right of the spiral, there are some white, angular, geometric shapes that look like fragments or pieces of a larger structure.

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

The diagram features a central title 'Declarative Name Binding for Type System Specifications' in a large, black, sans-serif font. Four blue arrows originate from the title: two point upwards towards the top text, and two point downwards towards the bottom text, creating a symmetrical layout.

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

A simple example

Syntax

integers $z \in \mathbb{Z} := \{\dots, -1, 0, 1, \dots\}$
 identifiers $x \in Id :=$ some countable set
 expressions $e \in Expr ::= z \mid e + e \mid \mathbf{fun}(x : t)\{ e \} \mid x \mid e \ e \mid \mathbf{let} \ x = e \ \mathbf{in} \ e$
 types $t \in Type ::= \mathbf{num} \mid t \rightarrow t$

Resolution Syntax Parameters

labels $l \in \mathcal{L} ::= P$
 relations $r \in \mathcal{R} ::= :$
 data terms $d \in \mathcal{D} ::= x_i : t$

Matching Declarations, Label Order and Data Order

$$\frac{x_i \simeq x_j}{(x_j : t) \in \text{DECL}(x_i)} \quad \$ <_l P \quad t_1 \leq_{\top} t_2 \quad \boxed{d \in \text{DECL}(x_i)} \quad \boxed{\hat{l} <_l \hat{l}} \quad \boxed{t \leq_{\top} t}$$

Typing Rules

$$\begin{array}{c}
 \text{(STLC-Num)} \frac{}{s \vdash z : \mathbf{num}} \quad \text{(STLC-Plus)} \frac{s \vdash e_1 : \mathbf{num} \quad s \vdash e_2 : \mathbf{num}}{s \vdash e_1 + e_2 : \mathbf{num}} \quad \boxed{\mathcal{G}, s \vdash e : t} \\
 \\
 \text{(STLC-Fun)} \frac{\nabla s_2 \quad s_2 \xrightarrow{P} s_1 \quad s_2 \xrightarrow{\cdot} \blacksquare x_i : t_1 \quad s_2 \vdash e : t_2}{s_1 \vdash \mathbf{fun}(x_i : t_1)\{ e \} : t_1 \rightarrow t_2} \\
 \\
 \text{(STLC-Id)} \frac{\text{DECL}(x_i), P^*, \leq_{\top}, <_l \vdash p : s \xrightarrow{\cdot} x_j : t}{s \vdash x_i : t} \quad \text{(STLC-App)} \frac{s \vdash e_1 : t_1 \rightarrow t_2 \quad s \vdash e_2 : t_1}{s \vdash e_1 \ e_2 : t_2} \\
 \\
 \text{(STLC-Let)} \frac{s_1 \vdash e_1 : t_1 \quad \nabla s_2 \quad s_2 \xrightarrow{P} s_1 \quad s_2 \xrightarrow{\cdot} \blacksquare x_i : t_1 \quad s_2 \vdash e_2 : t_2}{s_1 \vdash \mathbf{let} \ x_i = e_1 \ \mathbf{in} \ e_2 : t_2}
 \end{array}$$

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.

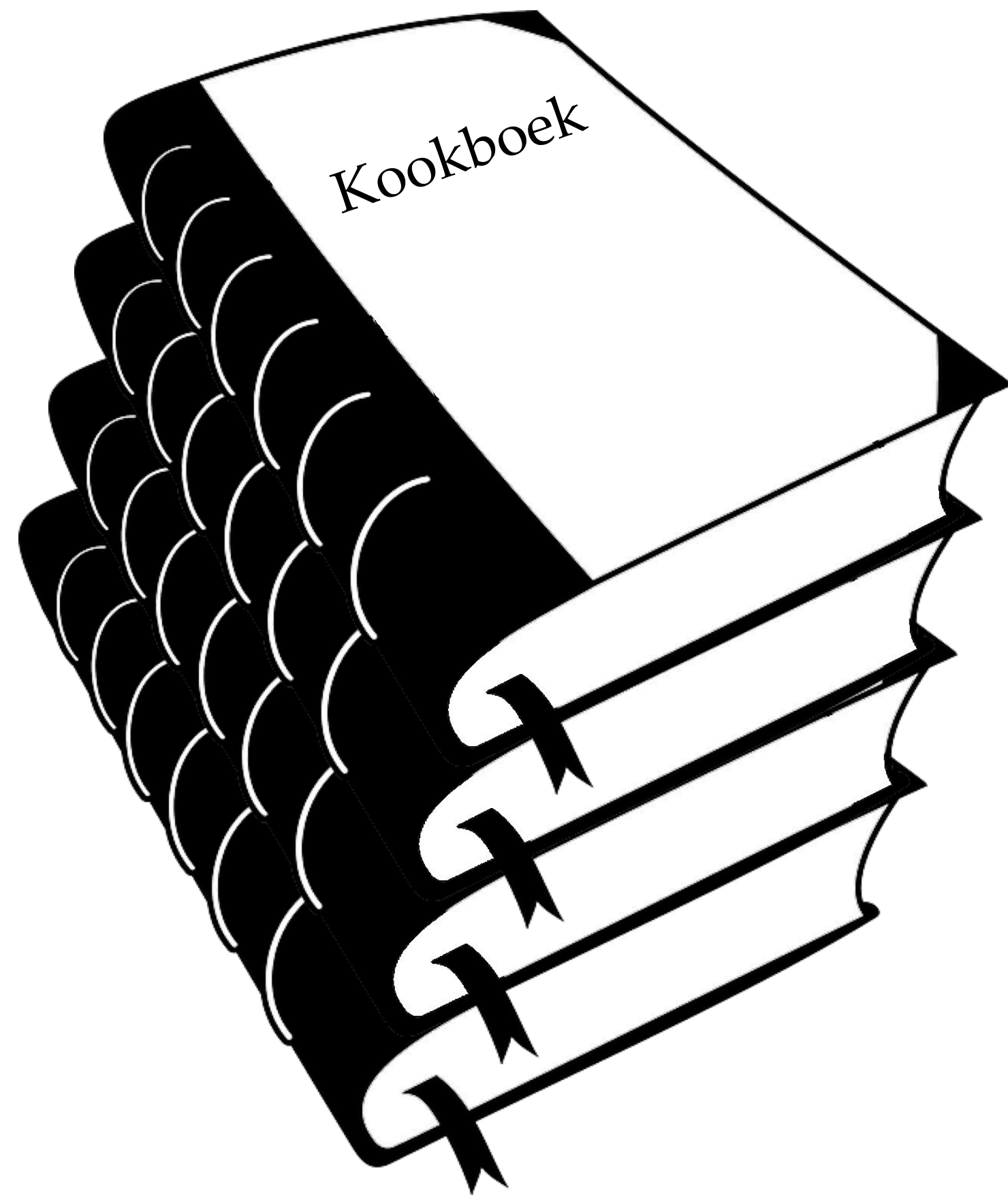
5. Remove the syrup from the heat, then snap most of the chocolate into the skillet.

RecipeChecker.exe

```
20 recipe.method.forEach((step, index) => {
21
22   // ... other cases ...
23
24   const re = /(\w+) the (\w+) into the (\w+)/g;
25   for (const match in step.matchAll(re)) {
26     const action = match[1];
27     const food_item = match[2];
28     const container = match[3];
29     if (!allowedActions.includes(action)) {
30       console.log(`step ${index+1}: unknown action ${action}`);
31     }
32     if (!findFoodItem(food_item, recipe.ingredients, recipe.method.slice(0, index))) {
33       console.log(`step ${index+1}: unknown ingredient ${food_item}`);
34     }
35     if (!findContainer(container, recipe.method.slice(0, index))) {
36       console.log(`step ${index+1}: unknown container ${container}`);
37     }
38   }
39
40   // ... other cases ...
41
42 });
43
44 function findFoodItem(
45   name: string,
46   ingredients: string[],
47   steps: string[],
48 ): boolean {
49   for (const step in steps) {
50     if (step.includes(`to a ${name}`)) { return true; }
51   }
52   for (const ingredient in ingredients) {
53     if (ingredient.includes(name)) { return true; }
54   }
55   return false;
56 }
57
58 function findContainer(
59   name: string,
60   steps: string[],
61 ): boolean {
62   for (const step in steps) {
63     if (step.includes(`into a ${name}`)) { return true; }
64   }
65   return false;
66 }
67 }
```



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!

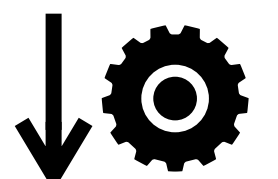
Boozy pears & chocolate

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.
- ...

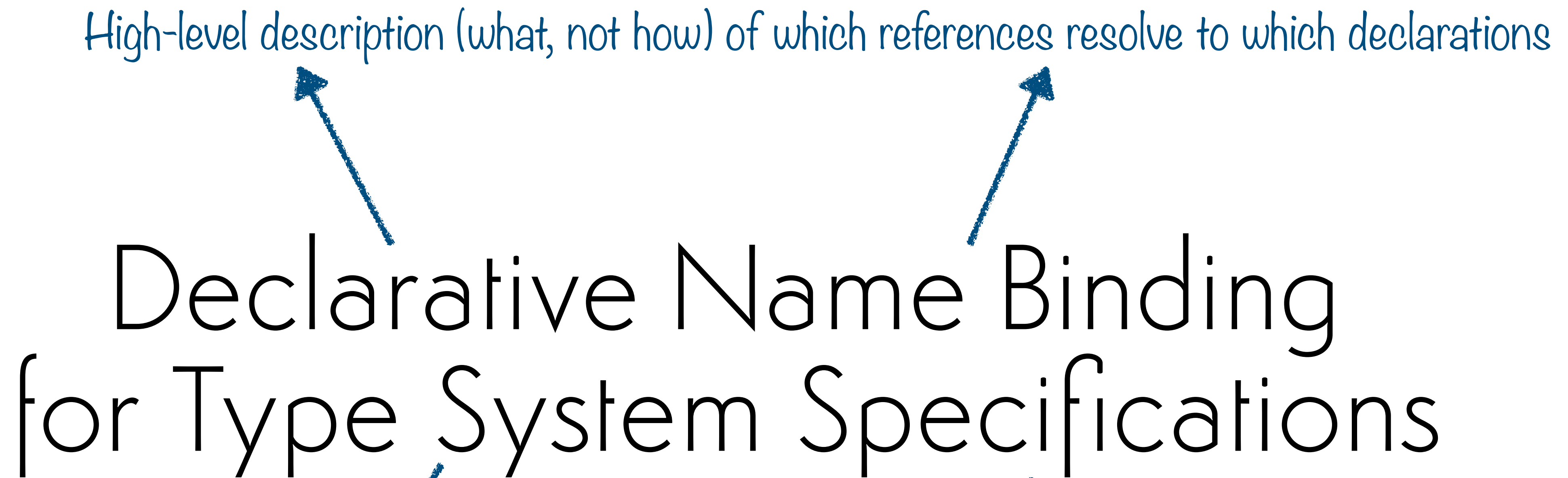


RecipeChecker.exe

```
10 recipeMethodForStep(step, index) = {
11   // ... other cases ...
12   // ... other cases ...
13 }
14
15 function findFoodItem(
16   name: string,
17   ingredients: string[],
18   steps: string[]) {
19   // ... other cases ...
20   // ... other cases ...
21 }
22
23 function findContainer(
24   name: string,
25   steps: string[]) {
26   // ... other cases ...
27   // ... other cases ...
28 }
29
30 return false
31 }
```



Declarative Name Binding for Type System Specifications



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

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