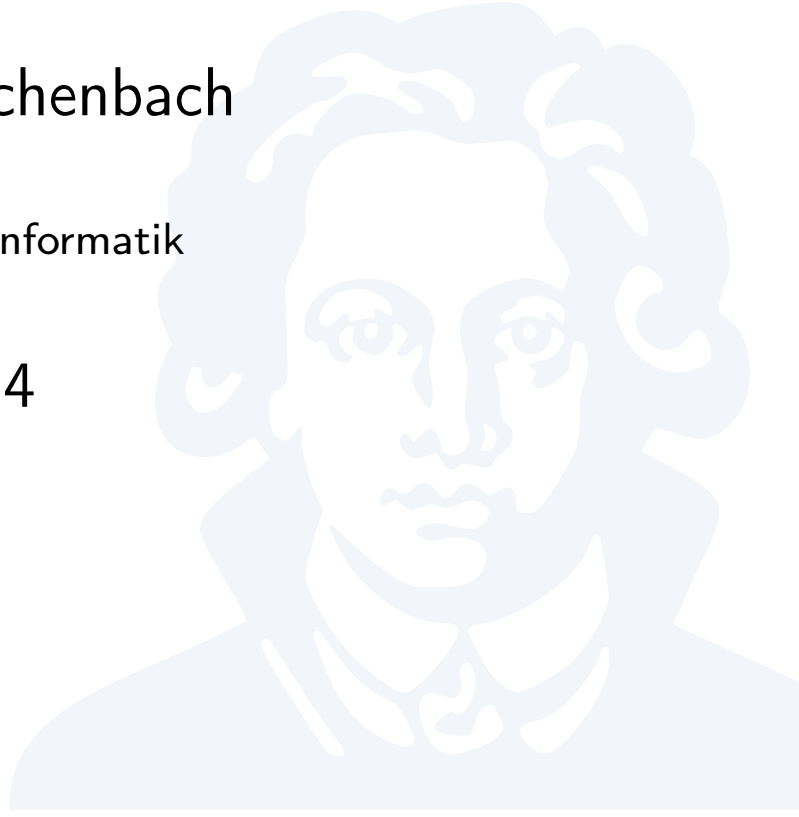
Foundations of Programming Languages

Names and Bindings

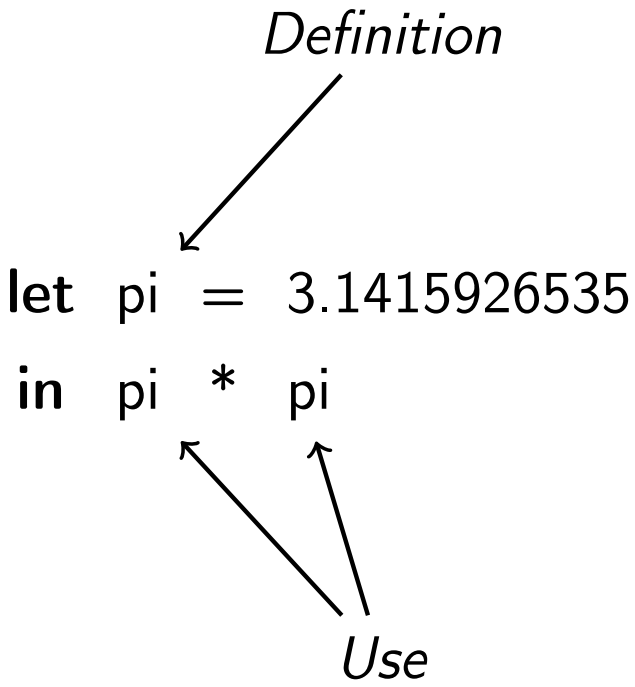
Prof. Dr. Christoph Reichenbach

Fachbereich 12 / Institut für Informatik

22. Oktober 2014



Names: Sharing Information



- Definition *binds* name to meaning
- *Meaning* can be a value, type, function, ...
- Valid names vary by language: `camelCase`, `foo_bar`, `lisp-name`, ...

name = identifier

Keywords and Reserved Words

Some name-shaped words have special meanings:

`if`, **`while`**, **`def`**, **`return`**, **`class`**, . . .

- **reserved word**: name with fixed purpose
- **keyword**: name with fixed purpose only in *some* contexts

FORTRAN

```
Integer Real   ! integer variable 'Real'
Real Integer   ! floating-point var. 'Integer'
if = 7         ! assign number to var. 'if'
```

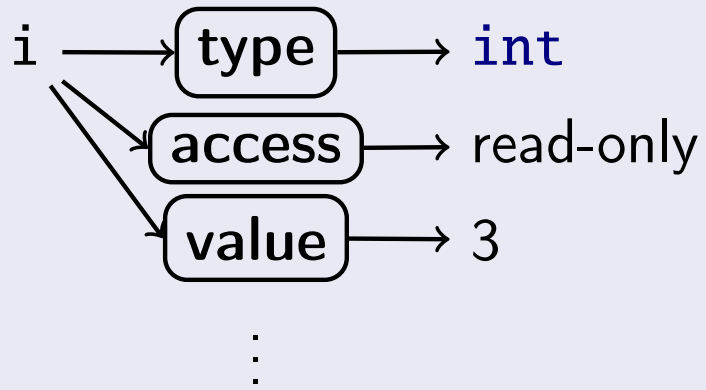
Newer languages prefer *reserved words*

Definitions and Bindings

Each definition introduces *bindings*:

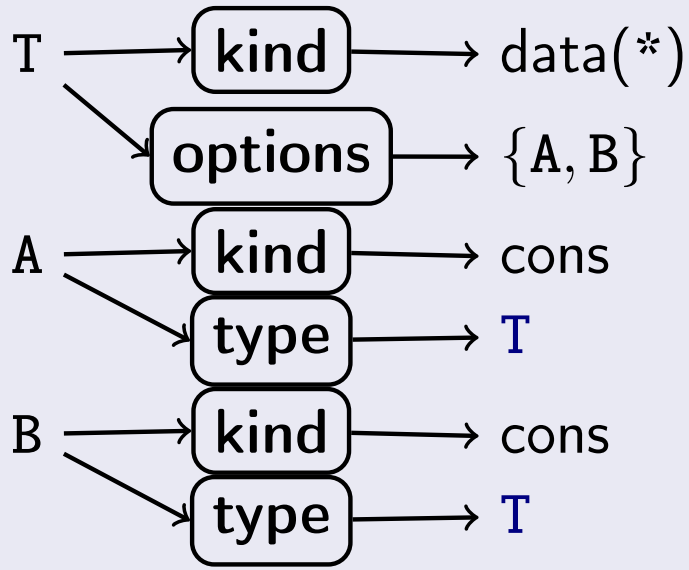
C

```
const int i = 3;
```



Haskell

```
data T = A | B
```



Bindings map names to *attributes*

Binding Time

- *language definition-time binding:*

  '+' binds to addition

- *static binding*:

Java

```
String s;
```

Type binding fixed (`String`) at compile time

- *dynamic binding*:

C

```
x = 2;
if (y > 1) x = 7;
```

Value binding changes at *runtime*

Further binding times possible

Summary

- *Names* allow sharing of information
- *Definitions* bind names to *attributes*
- Attributes contain many properties, such as types, values, access rights
- Definitions can create *bindings* at various times:
  - compile-time (*static binding*)
  - run-time (*dynamic binding*)

  ...