Universitetet i Oslo
Institutt for Informatikk
Research group for Reliable Systems (PSY)
Silvia Lizeth Tapia Tarifa, Riccardo Sieve

INF 5170: Models of Concurrency

Fall 2026 — **Group Session 6** — 21.10.2025

Topic: Types and Types for Concurrency

Exercise 1 Extend the following syntax for statements $\mathtt{s}$ such that the syntax also defines `return` statements, which enable to return the value defined by an expression. Given the syntax for types $\mathtt{T}$, define a typing rule for the `return` statements. You can assume that the typing rules given from the lecture already exist.

$$\begin{aligned}\mathtt{s} &::= \mathtt{v} \;=\; \mathtt{e};\mathtt{s} \mid \mathtt{T}\;\mathtt{v} \;=\; \mathtt{e};\;\mathtt{s} \mid \textbf{skip} \mid \textbf{if}(\mathtt{e})\{\mathtt{s}\}\mathtt{s}\\ \mathtt{e} &::= n \mid \mathit{true} \mid \mathit{false} \mid \mathtt{e}{+}\mathtt{e} \mid \mathtt{e} \wedge \mathtt{e} \mid \mathtt{e} \leq \mathtt{e} \mid \mathtt{v}\\ \mathtt{T} &::= \mathtt{Int} \mid \mathtt{Bool} \mid \mathtt{Unit}\end{aligned}$$

Solution: Updated syntax for statements $\mathtt{s}$:

$$\mathtt{s} ::= \mathtt{v} \;=\; \mathtt{e};\mathtt{s} \mid \mathtt{T}\;\mathtt{v} \;=\; \mathtt{e};\;\mathtt{s} \mid \textbf{skip} \mid \textbf{if}(\mathtt{e})\{\mathtt{s}\}\mathtt{s} \mid \textbf{return}\ \mathtt{e}$$

Typing rule for the `return` statements:

$$\frac{\Gamma \vdash \mathtt{e} : \mathtt{T}}{\Gamma \vdash \textbf{return}\ \mathtt{e} : \mathtt{Unit}}\ \textsf{return}$$

Exercise 2 We extend the syntax given in Example 1 such that we can perform function calls, where a function f maps an argument of type T_1 to type T_2, i.e., $f : \mathtt{T}_1 \mapsto \mathtt{T}_2$. An example of a function would be a successor function $\mathit{succ} : \mathtt{Int} \mapsto \mathtt{Int}$ that increases a given integer by one. Similar to variables, functions are also stored in the typing environment Γ. An example of Γ with a function would be $\Gamma = \{\mathit{succ} \mapsto \mathtt{Int} \mapsto \mathtt{Int}\}$ We update the syntax as follows:

$$\mathtt{e} ::= \; n \mid \mathit{true} \mid \mathit{false} \mid \mathtt{e}{+}\mathtt{e} \mid \mathtt{e} \wedge \mathtt{e} \mid \mathtt{e} \leq \mathtt{e} \mid \mathtt{v} \mid \mathtt{f}(\mathtt{e})$$

The type system of Example 1 is extended by the following typing rule:

$$\frac{\Gamma(\mathtt{f}) = \mathtt{T}_1 \mapsto \mathtt{T}_2 \qquad \Gamma \vdash \mathtt{e} : \mathtt{T}_1' \qquad \mathtt{T}_1' <: \mathtt{T}_1}{\Gamma \vdash \mathtt{f}(\mathtt{e}) : \mathtt{Unit}}\ \textsf{function-call}$$

Does this extension of the type still ensure type soundness considering also the typing rules in the lecture? If not, can you give an example that shows the violation of type soundness.

Solution: The soundness is violated as a function call is not a statement, but an expression. Therefore, the typing must be changed such the expression evaluate to the type given by the function.

$$\frac{\Gamma(\texttt{f}) = \texttt{T}_1 \mapsto \texttt{T}_2 \qquad \Gamma \vdash \texttt{e} : \texttt{T}_1' \qquad \texttt{T}_1' <: \texttt{T}_1}{\Gamma \vdash \texttt{f}(\texttt{e}) : \texttt{T}_2}\ \textsf{function-call}$$

Example showing the violation of the type soundness considering the following typing environment $\Gamma = \{\texttt{v} \mapsto \texttt{Int}, \mathit{succ} \mapsto \texttt{Int} \mapsto \texttt{Int}\}$:

$$\frac{\Gamma \vdash \mathit{succ}(\texttt{0}) : \texttt{Int} \qquad \dfrac{}{\texttt{Int} <: \texttt{Int}} \qquad \dfrac{}{\Gamma \vdash \textbf{skip} : \texttt{Unit}}\ \textsf{skip}}{\Gamma \vdash \texttt{v} = \mathit{succ}(\texttt{0}); \textbf{skip} : \texttt{Unit}}\ \textsf{assign}$$

The typing rule function-call cannot be applied to $\Gamma \vdash \mathit{succ}(\texttt{0}) : \texttt{Int}$ because $\mathit{succ}(0)$ is of type `Int`, not of type `Unit` as in the typing rule. Therefore, the progress condition of type soundness is not satisfied.

Exercise 3 Consider the Go-program given in Algorithm 1. Modify the program such that the channel types use (1) input/output modes, (2) linear types, and (3) usage types. The types should be defined as specific as possible.

Solution:

(1) Channels with typed input/output modes:

```
1 func add(inp chan? int, fin chan? bool, res chan! int) {...}
2
3 func provide(inp chan! int, fin chan! bool) {...}
4
5 func main() {
6     input := make(chan!? int)
7     finish := make(chan!? bool)
8     res := make(chan!? int)
9     ...}
```

(2) Channels with linear types:

```
1 func add(inp chan<?ω,!0> int, fin chan<?1,!0> bool, res chan<?0,!1> int) {...}
2
3 func provide(inp chan<?0,!ω> int, fin chan<?0,!1> bool) {...}
4
5 func main() {
6     input := make(chan<?ω,!ω> int)
7     finish := make(chan<?1,!1> bool)
8     res := make(chan<?1,!1> int)
9     ...}
```

(3) Channels with usage types:

```
1 func add(inp chan<?.?.0> int, fin chan<?.0> bool, res chan<!.0> int) {...}
2
3 func provide(inp chan<!.!.0> int, fin chan<!.0> bool) {...}
4
5 func main() {
6     input := make(chan<?.?.0 + !.!.0> int)
7     finish := make(chan<?.0 + !.0> bool)
8     res := make(chan<?.0 + !.0> int)
9     ...}
```

Exercise 4 Check if the Go-like program in Algorithm 2 is well-typed using typing trees. For type checking, consider the following typing environments using

1. linear types $\Gamma^1 = \{\texttt{inp} \mapsto \textbf{chan}_{?0,!1}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{?1,!1}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$, and
2. usage types $\Gamma^2 = \{\texttt{inp} \mapsto \textbf{chan}_{?.0+!.0}\ \texttt{int}, \texttt{sig} \mapsto \textbf{chan}_{?.0+!.0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$.

Solution: Solution for (1):

$\Gamma_1^1 = \{\texttt{inp} \mapsto \textbf{chan}_{?0,!1}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{?0,!1}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_2^1 = \Gamma_7^1 = \{\texttt{inp} \mapsto \textbf{chan}_{?0,!0}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{?1,!0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_3^1 = \Gamma_5^1 = \Gamma_6^1 = \Gamma_8^1 = \{\texttt{inp} \mapsto \textbf{chan}_{?0,!0}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{?0,!0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_4^1 = \{\texttt{inp} \mapsto \textbf{chan}_{?0,!0}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{?0,!1}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$$\dfrac{\dfrac{}{\Gamma^1 = \Gamma_1^1 + \Gamma_2^1} \quad \text{See typing tree } \textit{send} \quad \text{See typing tree } \textit{receive}}{\Gamma^1 \vdash \textit{go func}()\{\texttt{inp} \texttt{<-}1; \textit{sig} \texttt{<-}\textit{true};\ \texttt{skip}\}(); \texttt{v} = \texttt{<-sig};\ \texttt{skip} : \texttt{Unit}}\ (\text{L1})$$

Typing tree *send*:

$$\dfrac{\dfrac{\dfrac{}{\Gamma_4^1[\texttt{sig} \mapsto \textbf{chan}_{?0,!0}\ \texttt{int}] = \Gamma_5^1 + \Gamma_6^1} \quad \dfrac{}{\Gamma_4^1(\texttt{sig}) = \textbf{chan}_{?0,!1}\ \texttt{bool}} \quad \dfrac{\dfrac{}{\text{un}(\Gamma_5^1)}}{\Gamma_5^1 \vdash \textit{true} : \texttt{bool}}\ (\text{L5}) \quad \dfrac{\dfrac{}{\text{un}(\Gamma_6^1)}}{\Gamma_6^1 \vdash \texttt{skip} : \texttt{Unit}}\ (\text{L4})}{\Gamma_4^1 \vdash \texttt{sig} \texttt{<-}\textit{true};\ \texttt{skip} : \texttt{Unit}}\ (\text{L2}) \quad \dfrac{}{\Gamma_1^1[\texttt{inp} \mapsto \textbf{chan}_{?0,!0}\ \texttt{int}] = \Gamma_3^1 + \Gamma_4^1} \quad \dfrac{}{\Gamma_1^1(\texttt{inp}) = \textbf{chan}_{?0,!1}\ \texttt{int}} \quad \dfrac{\dfrac{}{\text{un}(\Gamma_3^1)}}{\Gamma_3^1 \vdash 1 : \texttt{int}}\ (\text{L6})}{\Gamma_1^1 \vdash \texttt{inp} \texttt{<-}1; \texttt{sig} \texttt{<-}\textit{true};\ \texttt{skip} : \texttt{Unit}}\ (\text{L2})$$

Typing tree *receive*:

$$\dfrac{\dfrac{}{\Gamma_2^1 = \Gamma_7^1 + \Gamma_8^1} \quad \dfrac{\dfrac{}{\Gamma_7^1(\texttt{sig}) = \texttt{chan}_{?1,!0}\texttt{bool}} \quad \dfrac{}{\text{un}(\Gamma_7^1[\texttt{skip} \mapsto \texttt{chan}_{?0,!0}])}}{\Gamma_7^1 \vdash \texttt{<-sig} : \texttt{bool}}\ (\text{L3}) \quad \dfrac{}{\Gamma_2^1(\texttt{v}) = \texttt{bool}} \quad \dfrac{\dfrac{}{\text{un}(\Gamma_8^1)}}{\Gamma_8^1 \vdash \texttt{skip} : \texttt{Unit}}\ (\text{L4})}{\Gamma_2^1 \vdash \texttt{v} = \texttt{<-sig};\ \texttt{skip} : \texttt{Unit}}\ (\text{L7})$$

Solution for (2):
Note that we skipped subtyping rules, since subtyping is also not considered for linear types.

$\Gamma_1^2 = \{\texttt{inp} \mapsto \textbf{chan}_{!.0}\ \texttt{int}, \texttt{sig} \mapsto \textbf{chan}_{!.0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_2^2 = \{\texttt{inp} \mapsto \textbf{chan}_{?.0}\ \texttt{int}, \texttt{sig} \mapsto \textbf{chan}_{?.0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_3^2 = \{\texttt{sig} \mapsto \textbf{chan}_{!.0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_4^2 = \{\texttt{inp} \mapsto \textbf{chan}_{0}\ \texttt{int},\ \texttt{v} \mapsto \texttt{bool}\}$

$\Gamma_5^2 = \{\texttt{inp} \mapsto \textbf{chan}_{?.0}\ \texttt{int},\ \texttt{v} \mapsto \texttt{bool}\}$

Note that rule U5 is a bit of a short cut, since it combines assign and receive. A more general set of typing rules would include separte rules for receive and assign.

$$\frac{\dfrac{}{\Gamma^2 = \Gamma_1^2 + \Gamma_2^2} \qquad \text{See typing tree } \textit{send} \qquad \text{See typing tree } \textit{receive}}{\Gamma^2 \vdash \textit{go func}()\{\texttt{inp} <\!\!-1; \textit{sig} <\!\!-\textit{true};\ \texttt{skip}\}(); \texttt{v} = <\!\!-\texttt{sig};\ \texttt{skip} : \texttt{Unit}}\ (\text{U4})$$

Typing tree *send*:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\textit{un}(\{\texttt{inp} \mapsto \textbf{chan}_0\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_0\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\})}\ (\text{U1})}{\Gamma_4^2 + \{\texttt{sig} \mapsto \textbf{chan}_0\ \texttt{bool}\} \vdash \texttt{skip}\colon \texttt{Unit}} \qquad \dfrac{}{\Gamma_4^2 \vdash \textit{true}\colon \texttt{bool}}\ (\text{U3})}{\{\texttt{inp} \mapsto \textbf{chan}_0\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_{!.0}\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\} \vdash \texttt{sig} <\!\!-\textit{true};\ \texttt{skip} : \texttt{Unit}}\ (\text{U6}) \qquad \dfrac{}{\Gamma_3^2 \vdash 1\colon \texttt{int}}\ (\text{U2})}{\Gamma_3^2 + \{\ \texttt{inp} \mapsto \textbf{chan}_{!.0}\ \texttt{int}\} \vdash \texttt{inp} <\!\!-1; \texttt{sig} <\!\!-\textit{true};\ \texttt{skip} : \texttt{Unit}}\ (\text{U6})$$

Typing tree *receive*:

$$\dfrac{\dfrac{\dfrac{\textit{un}(\{\texttt{inp} \mapsto \textbf{chan}_{?.0}\ \texttt{int},\ \texttt{sig} \mapsto \textbf{chan}_0\ \texttt{bool},\ \texttt{v} \mapsto \texttt{bool}\})}{\Gamma_5^2 + \{\texttt{sig} \mapsto \textbf{chan}_0\ \texttt{bool}\} \vdash \texttt{skip} : \texttt{Unit}}\ (\text{U1}) \qquad \dfrac{}{\Gamma_5^2(\texttt{v}) = \texttt{bool}}\ (\text{U3})}{\Gamma_5^2 + \{\ \texttt{sig} \mapsto \textbf{chan}_{?.0}\ \texttt{bool}\} \vdash \texttt{v} = <\!\!-\texttt{sig};\ \texttt{skip} : \texttt{Unit}}\ (\text{U5})$$

Exercise 5 Does the Go-program in Algorithm 3 with usage types type-check? If not, explain why and how the program can be corrected? Annotate the program with the updated usage types.

Solution: The program does not type-check as channel `a` is never read. Note that splitting nested channels is undefined.

```
1   func main() {
2   b := make(chan<?.0 + !.0> chan<?.0 + !.0> int)
3   a := make(chan<?.0 + !.0> int)
4   go func() {   // b ↦ chan<!.0> chan<?.0> int (solution 1) or ↦ chan<!.0> chan<0> in (solution 2)
5       b <− a // b ↦ chan<0> chan<?.0> int. (solution 1) or ↦ chan<!.0> chan<0> in (solution 2)
6       <−a // solution 1: a ↦ chan<0> int
7   }()
8   // b ↦ chan<?.0> chan<?.0> int, a ↦ chan<?.0> int
9   go func(c chan<!.0> int) { // c = a ↦ chan<!.0> int
10      c <− 1 // c = a ↦ chan<0> int
11  }(<−b) // b ↦ chan<0> chan<!.0> int
12
13  <−a // solution 2: a ↦ chan<0> int
14  }
15
```

Algorithm 1 Sum up integer from given input stream.

```
1  func add(inp chan int, fin chan bool, res chan int) {
2      sum := 0
3      for { select {
4          case num := <−inp: { sum = sum + num }
5          case <−fin: { res <− sum }
6          }
7      }
8  }
9
10 func provide(inp chan int, fin chan bool) {
11     inp <− 9
12     inp <− 3
13     fin <− true
14 }
15
16 func main() {
17     input := make(chan int)
18     finish := make(chan bool)
19     res := make(chan int)
20     go add(input, finish , res)
21     go provide(input, finish )
22     <−res
23 }
```

Algorithm 2 Snippet of a Go-like program using linear types.

```
1          go func() { inp <− 1; sig <− true; skip }()
2          v = <− sig; skip
3
```

Algorithm 3 Go-program with usage types

```
1      func main() {
2      b := make(chan<?.0 + !.0> chan<?.0 + !.0> int)
3      a := make(chan<?.0 + !.0> int)
4      go func() {
5          b <− a
6      }()
7      go func(c chan<!.0> int) {
8          c <− 1
9      }(<−b)
10 }
```

Appendix

Typing rules for a linear type system:

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \qquad \Gamma_1 \vdash \mathtt{s_1} : \mathtt{Unit} \qquad \Gamma_2 \vdash \mathtt{s_2} : \mathtt{Unit}}{\Gamma \vdash \textbf{go}\ \mathtt{func()\{s_1\}();\ s_2} : \mathtt{Unit}}\ \text{L-parallel (L1)}$$

$$\frac{\Gamma[\mathtt{c} \mapsto \textbf{chan}_{?n,!0}\ T] = \Gamma_1 + \Gamma_2 \qquad \Gamma(\mathtt{c}) = \textbf{chan}_{?n,!1}\ T \qquad \Gamma_1 \vdash \mathtt{s} : \mathtt{Unit} \qquad \Gamma_2 \vdash \mathtt{e} : T}{\Gamma \vdash \mathtt{c}\ <\!\!-\ \mathtt{e};\ \mathtt{s} : \mathtt{Unit}}\ \text{L-send (L2)}$$

$$\frac{\Gamma(\mathtt{v}) = \textbf{chan}_{?1,!0}\ T \qquad \mathsf{un}(\Gamma[\mathtt{v} \mapsto \textbf{chan}_{?0,!0}\ T])}{\Gamma \vdash \leftarrow\!\mathtt{v} : T}\ \text{L-receive (L3)}$$

$$\frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \textbf{skip} : \mathtt{Unit}}\ \text{L-skip (L4)} \qquad \frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \mathit{true} : \mathtt{bool}}\ \text{L-true (L5)}$$

$$\frac{\mathsf{un}(\Gamma)}{\Gamma \vdash n : \mathtt{int}}\ \text{L-int (L6)}$$

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \qquad \Gamma_1 \vdash \mathtt{e} : T \qquad \Gamma(\mathtt{v}) = T \qquad \Gamma_2 \vdash \mathtt{s} : \mathtt{Unit}}{\Gamma \vdash \mathtt{v}\ =\ \mathtt{e};\ \mathtt{s} : \mathtt{Unit}}\ \text{L-assign (L7)}$$

Typing rules for a usage type system:

$$\frac{\mathsf{un}(\Gamma)}{\Gamma \vdash \textbf{skip} : \mathtt{Unit}}\ \text{U-skip (U1)} \qquad \frac{}{\Gamma \vdash n : \mathtt{int}}\ \text{U-int (U2)}$$

$$\frac{}{\Gamma \vdash \mathit{true} : \mathtt{bool}}\ \text{U-true (U3)}$$

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \qquad \Gamma_1 \vdash \mathtt{s_1} : \mathtt{Unit} \qquad \Gamma_2 \vdash \mathtt{s_2} : \mathtt{Unit}}{\Gamma \vdash \textbf{go}\ \textbf{func}()\{\ \mathtt{s_1}\}();\ \mathtt{s_2} : \mathtt{Unit}}\ \text{U-parallel (U4)}$$

$$\frac{\Gamma + \{c \mapsto \mathtt{chan_U}\ T\} \vdash \mathtt{s}\colon \mathtt{Unit} \qquad \Gamma \vdash \mathtt{v}\colon T' \qquad T <\!: T'}{\Gamma + \{c \mapsto \mathtt{chan_{?.U}}\ T\} \vdash \mathtt{v} =\!<\!\!-\mathtt{c};\ \mathtt{s}\colon \mathtt{Unit}}\ \text{U-assign-receive (U5)}$$

$$\frac{\Gamma + \{c \mapsto \mathtt{chan_U}\ T\} \vdash \mathtt{s}\colon \mathtt{Unit} \qquad \Gamma \vdash \mathtt{e}\colon T' \qquad T' <\!: T}{\Gamma + \{c \mapsto \mathtt{chan_{!.U}}\ T\} \vdash \mathtt{c}\ <\!\!-\mathtt{e};\ \mathtt{s}\colon \mathtt{Unit}}\ \text{U-send (U6)}$$