Language Transformations in the Classroom

Matteo Cimini (UMass Lowell)
Benjamin Mourad (UMass Lowell)

23rd August, EXPRESS/SOS 2021

Typical undergraduate PL course:

"Principles of Programming Languages"
"Organization of Programming Languages"
…

Syntax (BNF)
Variables
Functions
Scoping mechanisms
Types
Subtyping
Polymorphisms
Abstract machines
Memory management
Concurrency
…

Typical undergraduate PL course:

"Principles of Programming Languages"
"Organization of Programming Languages"
...

Syntax (BNF)
Variables
Functions
Scoping mechanisms
Types
Subtyping
Polymorphisms
Abstract machines
Memory management
Concurrency
...

Observation:

Some features are a "transformation" of a base language

Subtyping as a Modification of a Base Language

$$\frac{\Gamma \vdash e_1 : T_1 \to T_2 \qquad \Gamma \vdash e_2 : T_1}{\Gamma \vdash e_1\ e_2 : T_2}\ (\text{T-APP}) \quad \Longrightarrow \quad \frac{\Gamma \vdash e_1 : T_{11} \to T_2 \qquad \Gamma \vdash e_2 : T_{12} \qquad T_{12} <: T_{11}}{\Gamma \vdash e_1\ e_2 : T_2}\ (\text{T-APP'})$$

(T-APP) rejects $((\lambda x : \texttt{float}.x)\ 3)$, where $\emptyset \vdash 3 : \texttt{int}$

(T-APP'):

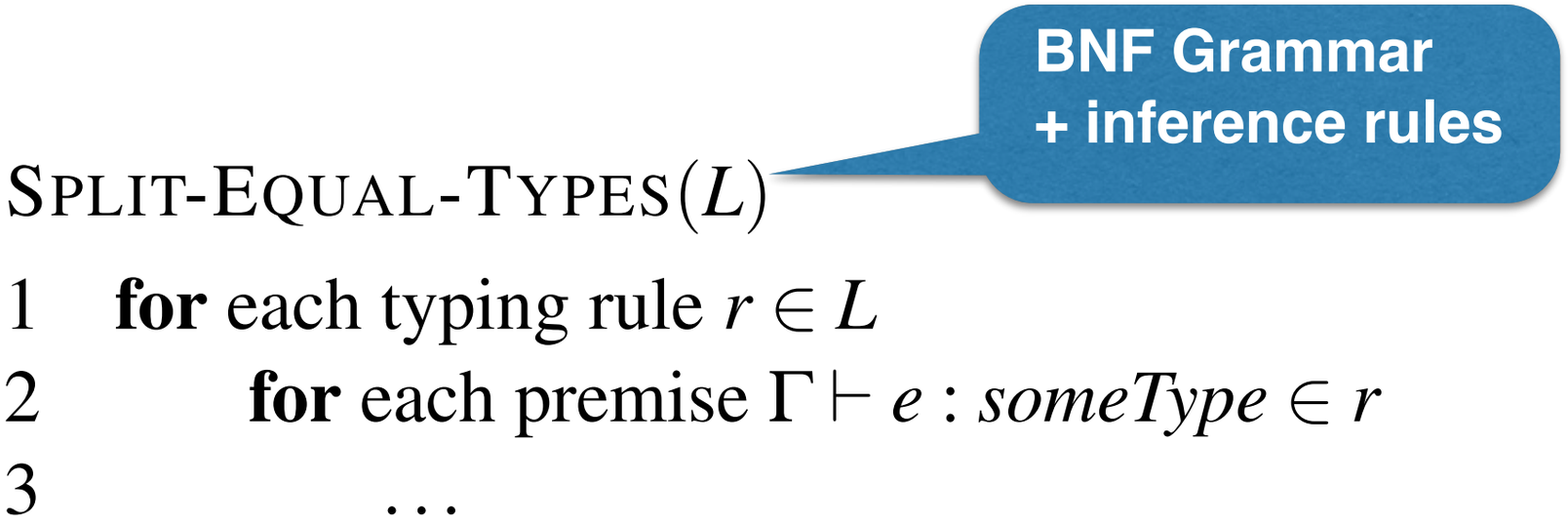
Step 1: ***Split equal types***

Step 2: ***Relate new variables by subtyping according to variance***

Language Transformations [1,2]

(T-APP′):

Step 1: *Split equal types*
Step 2: *Relate new variables by subtyping according to variance*



[1] ***A Calculus for Language Transformations.***
Benjamin Mourad, Matteo Cimini. SOFSEM 2020
[2] ***System Description: Lang-n-Change - A Tool for Transforming Languages.***
Benjamin Mourad, Matteo Cimini. FLOPS 2020.

Benefits of Language Transformations

- **Apply to multiple languages**
- **Highlight the insights behind a feature**

Our Thesis:

Using language transformations for teaching PL features, in addition to the planned material, can be beneficial for students to deepen their understanding of the features being taught

Our Contributions

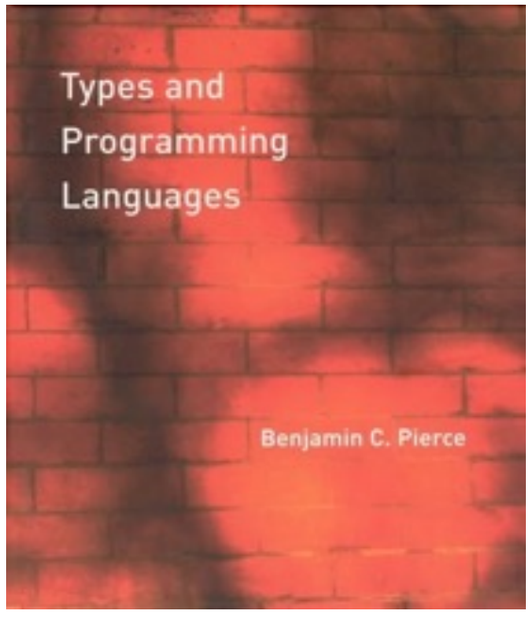
- Set forth the thesis that Language Transformations can be beneficial in the classroom.
- 2 instances of an undergraduate PL courses. (55 students) Language Transformations to help teaching
  - Subtyping
  - CK machine
  - We describe the study that we have conducted, material that we have taught, exam that we have submitted to students, and the results of the study

Ultimately:

We do not draw any general conclusion about our thesis
We offer our paper to show the kind of studies we are conducting
And to inspire similar studies (for, or against, our thesis).

Our Undergraduate Course

Textbook:



Language Specifications

Type	T	::=	$T \to T \mid B$
Expression	e	::=	$x \mid \lambda x{:}T.\, e \mid e\ e$
Value	v	::=	$\lambda x{:}T.\, e$
Evaluation Context	E	::=	$[\cdot] \mid E\ e \mid v\ E$
Type Environment	Γ	::=	$\emptyset \mid \Gamma, x : T$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\Gamma, x : T_1 \vdash e : T_2}{\Gamma \vdash (\lambda x{:}T_1.\, e) : T_1 \to T_2} \qquad \frac{\Gamma \vdash e_1 : T_{11} \to T_{12} \qquad \Gamma \vdash e_2 : T_{11}}{\Gamma \vdash e_1\ e_2 : T_{12}}\ \text{(T-APP)}$$

$$((\lambda x{:}T.e)\ v) \longrightarrow e[v/x] \qquad \frac{e \longrightarrow e'}{E[e] \longrightarrow E[e']}$$

Language Transformations

- We use pseudo-code (for, while, assignments, etc)
- Language specific operations: **L.rules**, **r.premises**, **r.conclusion**

Subtyping - Split Equal Types

SPLIT-EQUAL-TYPES(P)

1 $newPremises = \emptyset, varmap = \emptyset$
2 **for** each $p \in P$
3 **if** p is of the form $\Gamma \vdash e : someType$
4 **for** each $T \in someType$ s.t. T appears more than once in P
5 $T' =$ FRESH(P)
6 $p = p$ where T' replaces T in $someType$
7 $varmap(T) = varmap(T) \cup \{T'\}$
8 $newPremises = newPremises \cup \{p\}$
9 **return** $(newPremises, varmap)$

Recall

(T-APP)

$$\frac{\Gamma \vdash e_1 : T_1 \to T_2 \quad \Gamma \vdash e_2 : T_1}{\Gamma \vdash e_1\ e_2 : T_2}$$

Applied to **(T-APP)**

Input: $P = \{\Gamma \vdash e_1 : T_1 \to T_2, \Gamma \vdash e_2 : T_1\}$
Output = $(newPremises, varmap)$ where

$$\begin{aligned} newPremises &= \{\Gamma \vdash e_1 : T_{11} \to T_2, \Gamma \vdash e_2 : T_{12}\} \\ varmap &= \{T_1 \mapsto \{T_{11}, T_{12}\}\} \end{aligned}$$

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES($r.premises$)
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \ \vee \ T_2 \ \vee \ \ldots \ \vee \ T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES($r.premises$)
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \vee T_2 \vee \ldots \vee T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Select each typing rule

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES$(r.premises)$
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \vee T_2 \vee \ldots \vee T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Calls SPLIT-EQUAL-TYPES

Obtains

$$newPremises = \{\Gamma \vdash e_1 : T_{11} \to T_2, \Gamma \vdash e_2 : T_{12}\}$$
$$varmap = \{T_1 \mapsto \{T_{11}, T_{12}\}\}$$

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES($r.premises$)
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \vee T_2 \vee \ldots \vee T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Case where the type is invariant:

(T-ASSIGN)

$$\frac{\Gamma \vdash e_1 : \texttt{Ref}\ T \qquad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 := e_2 : \texttt{unitType}} \quad \Longrightarrow \quad \frac{\Gamma \vdash e_1 : \texttt{Ref}\ T_1 \qquad \Gamma \vdash e_2 : T_2 \qquad T_1 = T_2}{\Gamma \vdash e_1 := e_2 : \texttt{unitType}}$$

(T-ASSIGN')

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES($r.premises$)
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \vee T_2 \vee \ldots \vee T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Case where the type is contravariant:

(T-APP)

$$\frac{\Gamma \vdash e_1 : T_1 \to T_2 \qquad \Gamma \vdash e_2 : T_1}{\Gamma \vdash e_1\ e_2 : T_2} \quad \Longrightarrow \quad \frac{\Gamma \vdash e_1 : T_{11} \to T_2 \qquad \Gamma \vdash e_2 : T_{12} \qquad T_{12} <: T_{11}}{\Gamma \vdash e_1\ e_2 : T_2}$$

(T-APP')

Subtyping - Relate New Vars with Subtyping

ADD-SUBTYPING(L)

1 **for** each rule $r \in L.rules$ s.t. $r.conclusion$ is of the form $\Gamma \vdash e : someType$
2 $(newPremises, varmap) =$ SPLIT-EQUAL-TYPES($r.premises$)
3 **for** each mapping $(T \mapsto setOfNewVars)$ in $varmap$
4 **if** there exists a type in $setOfNewVars$ that is invariant in $newPremises$
5 **for** each $T_1, T_2 \in setOfNewVars$
6 $newPremises = newPremises \cup \{T_1 = T_2\}$
7 **elseif** there is exactly one type T' in $setOfNewVars$ that is contravariant in $newPremises$
8 **for** each $T_{new} \in (setOfNewVars \setminus T')$
9 $newPremises = newPremises \cup \{T_{new} <: T'\}$
10 **elseif** none in $setOfNewVars$ is contravariant or invariant in $newPremises$
11 say that $setOfNewVars = \{T_1, T_2, \ldots, T_n\}$
12 $newPremises = newPremises \cup \{T = T_1 \vee T_2 \vee \ldots \vee T_n\}$
13 **else** *error*
14 $r.premises = newPremises$

Case where the types are peer:

(T-IF)

$$\frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : T \qquad \Gamma \vdash e_3 : T}{\Gamma \vdash (if\ e_1\ e_2\ e_3) : T}$$

$\Longrightarrow$

(T-IF')

$$\frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : T_1 \qquad \Gamma \vdash e_3 : T_2 \qquad T = T_1 \vee T_2}{\Gamma \vdash (if\ e_1\ e_2\ e_3) : T}$$

Subtyping - Relate New Vars with Subtyping

$$newPremises = newPremises \cup \{T = T_1 \lor T_2 \lor \ \ldots \ \lor T_n\}$$

else *error*

$$r.premises = newPremises$$

Example:
one type is used in multiple contravariant positions

T1 T2 **T3 T4**

$$\frac{\Gamma \vdash e_1 : T \to (T \to \texttt{Bool}) \qquad \Gamma \vdash e_2 : T \times T}{\Gamma \vdash \texttt{app2}\ e_1\ e_2 : \texttt{Bool}}$$

$$\texttt{app2}\ e_1\ e_2 \longrightarrow ((e_1\ (\texttt{fst}\ e_2))\ (\texttt{snd}\ e_2)) \qquad or \qquad \texttt{app2}\ e_1\ e_2 \longrightarrow ((e_1\ (\texttt{snd}\ e_2))\ (\texttt{fst}\ e_2))$$

T3 <: T1
T4 <: T2

T4 <: T1
T3 <: T2

CK Machine

Problem with reduction semantics: (recomputing context)

$$(\texttt{if}\ (\texttt{hd}\ [\texttt{f} \wedge \boxed{((\lambda x.x)\ \texttt{t})}, \texttt{t}]))\ e_1\ e_2) \longrightarrow (\texttt{if}\ (\texttt{hd}\ [\texttt{f} \wedge \texttt{t}, \texttt{t}]))\ e_1\ e_2)$$

$$(\texttt{if}\ (\texttt{hd}\ [\boxed{\texttt{f} \wedge \texttt{t}}, \texttt{t}]))\ e_1\ e_2) \longrightarrow \ldots$$

Solution: CK Machine (Carry continuation at run-time)

$$\text{Continuation}\ \ k \quad ::= \quad \texttt{mt} \mid (\texttt{app}_1\ e\ k) \mid (\texttt{app}_2\ v\ k)$$

$$(\texttt{app}\ e\ e_2), k \longrightarrow e, (\texttt{app}_1\ e_2\ k) \qquad \texttt{Start}$$

$$v, (\texttt{app}_1\ e\ k) \longrightarrow e, (\texttt{app}_2\ v\ k) \qquad \texttt{Order}$$

$$v, (\texttt{app}_2\ (\lambda x.e)\ k) \longrightarrow e[v/x], k \qquad \texttt{Computation}$$

CK Machine - Generating Continuation Grammar

CK-GENERATE-GRAMMAR$(EvalCtx)$

1 create grammar category `Continuation`, and add grammar item `mt` to it
2 **for** each $(\mathtt{op}\ t_1 \ldots t_n) \in EvalCtx$
3 **if** $t_i = E$
4 add grammar item $(\mathtt{op}_i\ (t_1 \ldots t_n \text{ minus } E)\ k)$ to `Continuation`

Input: Evaluation Context $E ::= [\cdot] \mid E\ e \mid v\ E$

(app E e) ==> (app1 e k)
(app v E) ==> (app2 v k)

Output: Continuation $k ::= \mathtt{mt} \mid (\mathtt{app}_1\ e\ k) \mid (\mathtt{app}_2\ v\ k)$

CK Machine - Generating Continuation Grammar

Recall: $(\texttt{app}\ e\ e_2), k \longrightarrow e, (\texttt{app}_1\ e_2\ k)$ **(Start)**

CK-GENERATE-START$(\mathit{Continuations})$

1 find $(\texttt{op}_i\ t_1 \ldots t_n\ k) \in \mathit{Continuations}$ with no v
2 add reduction rule $(\texttt{op}\ t_1 \ldots\ e\ \ldots t_n), k \longrightarrow e, (\texttt{op}_i\ t_1 \ldots t_n\ k)$
e appears at position i in op.

Input: Continuation k $::=$ $\texttt{mt} \mid (\texttt{app}_1\ e\ k) \mid (\texttt{app}_2\ v\ k)$

Output: $(\texttt{app}\ e\ e_2), k \longrightarrow e, (\texttt{app}_1\ e_2\ k)$

Final Exam

Goal:

To see whether, if presented with a language with unusual operators, students would be able to add subtyping to it, and derive its CK machine.

Toy Language langFunny:

Simply typed lambda-calculus with pairs and list

+ doublyApply: $\textbf{doublyApply}\ v_1\ v_2\ v_3\ v_4 \longrightarrow \langle (v_2\ (v_1\ v_3)), (v_1\ (v_2\ v_4)) \rangle$

+ addToPairAsList: $\textbf{addToPairAsList}\ v_1\ \langle v_2, v_3 \rangle \longrightarrow [v_1, v_2, v_3]$

doublyApply - Specification given to students

$$\text{Evaluation Context } E ::= \ldots \mid (\textbf{doublyApply } E\ e\ e\ e) \mid (\textbf{doublyApply } v\ E\ e\ e) \mid (\textbf{doublyApply } v\ v\ E\ e) \mid (\textbf{doublyApply } v\ v\ v\ E)$$

$$\frac{\Gamma \vdash e_1 : T_1 \to T_2 \qquad \Gamma \vdash e_2 : T_2 \to T_1 \qquad \Gamma \vdash e_3 : T_1 \qquad \Gamma \vdash e_4 : T_2}{\Gamma \vdash (\textbf{doublyApply } e_1\ e_2\ e_3\ e_4) : T_2 \times T_1}$$

doublyApply - Solution requested

only premises

$$\frac{\Gamma \vdash e_1 : T_1 \to T_2' \qquad \Gamma \vdash e_2 : T_2 \to T_1' \qquad \Gamma \vdash e_3 : T_1'' \qquad \Gamma \vdash e_4 : T_2'' \qquad T_1' <: T_1 \qquad T_1'' <: T_1 \qquad T_2' <: T_2 \qquad T_2'' <: T_2}{}$$

$$(\textbf{doublyApply } e_1\ e_2\ e_3\ e_4), k \longrightarrow e_1, (\textbf{doublyApply}_1\ e_2\ e_3\ e_4\ k) \quad \texttt{Start}$$

$$v_1, (\textbf{doublyApply}_1\ e_2\ e_3\ e_4\ k) \longrightarrow e_2, (\textbf{doublyApply}_2\ v_1\ e_3\ e_4\ k) \quad \texttt{Order}$$

$$v_2, (\textbf{doublyApply}_2\ v_1\ e_3\ e_4\ k) \longrightarrow e_3, (\textbf{doublyApply}_3\ v_1\ v_2\ e_4\ k) \quad \texttt{Order}$$

$$v_3, (\textbf{doublyApply}_3\ v_1\ v_2\ e_4\ k) \longrightarrow e_4, (\textbf{doublyApply}_4\ v_1\ v_2\ v_3\ k) \quad \texttt{Order}$$

$$v_4, (\textbf{doublyApply}_4\ v_1\ v_2\ v_3\ k) \longrightarrow \langle (v_2\ (v_1\ v_3)), (v_1\ (v_2\ v_4)) \rangle, k \quad \texttt{Computation}$$

addToPairAsList - Specification given to students

$$\frac{\Gamma \vdash e_1 : T \qquad \Gamma \vdash e_2 : T \times T}{\Gamma \vdash (\textbf{addToPairAsList}\ e_1\ e_2) : \mathit{List}\ T}$$

addToPairAsList - Solution requested

$$\frac{\Gamma \vdash e_1 : T' \qquad \Gamma \vdash e_2 : T'' \times T''' \qquad T = T' \vee T'' \vee T'''}{\Gamma \vdash (\textbf{addToPairAsList}\ e_1\ e_2) : \mathit{List}\ T}$$

Results

- 2 iterations of the course
- 55 students

	Correct	Partially Correct	Partially Incorrect	Incorrect/Missing
Subtyping of **doublyApply**	15	11	15	14
Subtyping of **addToPairAsList**	22	10	11	12
CK for **doublyApply**	18	13	10	14

Survey

The language transformation algorithm for adding subtyping to languages helped me understand subtyping better

- Strongly Agree
- Somewhat Agree
- Neither Agree nor Disagree
- Somewhat Disagree
- Strongly Disagree
- N/A or do not remember

and

The language transformation algorithm for adding subtyping to languages helped me add subtyping to the language at hand during the exam

and the corresponding statements for the CK machine

Unfortunately: no participation to the survey

Conclusions

We have set forth the thesis that language transformations can be beneficial in the classroom, we have experimented with them, but we do not draw any general conclusion from our study.

We offer our paper to show the kind of studies we are conducting, and to inspire similar studies (for, or against, our thesis).

Future Work

- **Felleisen and Friedman's full CEK, SECD, Krivine.**
- **Bounded polymorphism, recursive subtyping.**
- **Big- to small-step semantics conversions, and viceversa**
- **Generating Milner's type inference procedures**

Thank you!