UNIVERSITÉ
CÔTE D'AZUR

Paradigmes et Interprétation

Sous-typage

Julien Provillard
julien.provillard@univ-cotedazur.fr

UNIVERSITÉ
CÔTE D'AZUR

TYPAGE DES ENREGISTREMENTS

UNIVERSITÉ CÔTE D'AZUR

Grammaire

```
<Exp> ::= <Number>
        | <Symbol>
        | {+ <Exp> <Exp>}
        | {* <Exp> <Exp>}
        | {lambda {[<Symbol> : <Type>]} <Exp>}
        | {<Exp> <Exp>}
        | {record [<Symbol> <Exp>]*}
        | {get <Exp> <Symbol>}
        | {set <Exp> <Symbol> <Exp>}

<Type> ::= num
         | bool
         | (<Type> -> <Type>)
         | {[<Symbol> : <Type>]*}
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{record
  [x {+ 1 2}]
  [y {* 3 4}]}
```

est

```
{[x : num] [y : num]}
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{get {record
        [x {+ 1 2}]
        [y {* 3 4}]}
     x}
```

est

```
num
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{get
  {get
    {record
      [p {record
           [x {+ 1 2}]
           [y {* 3 4}]}]}
    p}
  x}
```

est

```
num
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{set {record
        [x {+ 1 2}]
        [y {* 3 4}]}
     x
     5}
```

est le même que la valeur produite

```
{record
  [x 5]
  [y 12]}
```

soit

```
{[x : num] [y : num]}
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{{lambda {[r : {[x : num] [y : num]}]}
   {+ {get r x} {get r y}}}
 {record
   [x {+ 1 2}]
   [y {* 3 4}]}}
```

est

```
num
```

UNIVERSITÉ CÔTE D'AZUR

Typage des enregistrements

- Le type de

```
{lambda {[r : {[x : num] [y : num]}]}
  {set r x {get r y}}}
```

est

```
({[x : num] [y : num]} -> {[x : num] [y : num]})
```

UNIVERSITÉ CÔTE D'AZUR

Règles de typage

$$\frac{\Gamma \vdash expr_1 : \tau_1 \quad \ldots \quad \Gamma \vdash expr_n : \tau_n \qquad \tau = \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\}}{\Gamma \vdash \{record\ [x_1\ expr_1] \ldots [x_n\ expr_n]\} : \tau}$$

$$\frac{\Gamma \vdash expr : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \qquad x = x_i}{\Gamma \vdash \{get\ expr\ x\} : \tau_i}$$

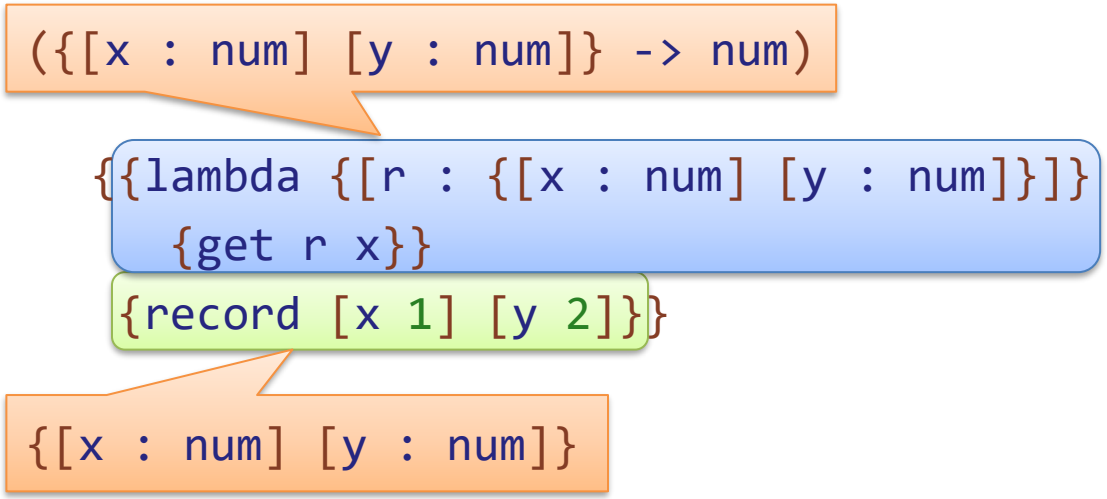
$$\frac{\Gamma \vdash expr_1 : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \qquad x = x_i \qquad \Gamma \vdash expr_2 : \tau_i}{\Gamma \vdash \{set\ expr_1\ x\ expr_2\} : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\}}$$

UNIVERSITÉ
CÔTE D'AZUR

SOUS-TYPAGE

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs



L'expression a pour type num.

$$\frac{\Gamma \vdash expr_1 : (\tau_1 \text{ -> } \tau_2) \qquad \Gamma \vdash expr_2 : \tau_1}{\Gamma \vdash \{expr_1\ expr_2\} : \tau_2}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs

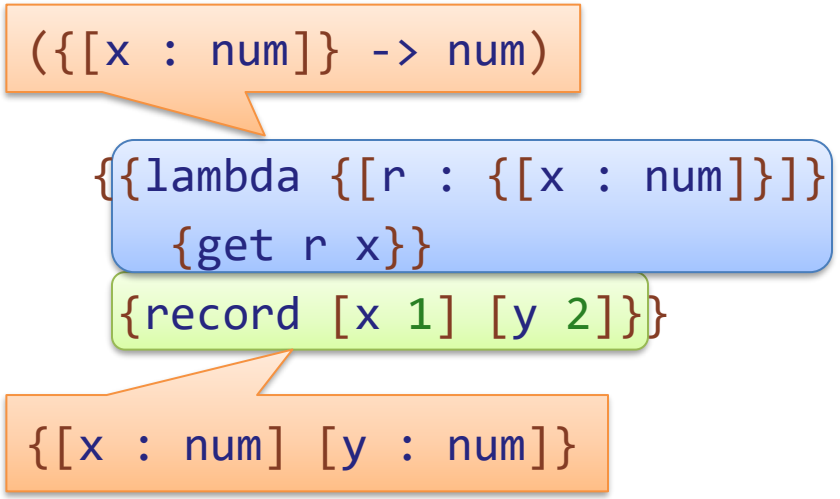
```
({[x : num] [y : num]} -> num)

{{lambda {[r : {[x : num] [y : num]}]}
   {get r x}}
 {record [y 1] [x 2]}}

{[y : num] [x : num]}
```

Non-typable : L'ordre des champs ne correspond pas.

$$\frac{\Gamma \vdash expr_1 : (\tau_1 \text{ -> } \tau_2) \qquad \Gamma \vdash expr_2 : \tau_1}{\Gamma \vdash \{expr_1\ expr_2\} : \tau_2}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs



Non-typable : L'argument a des champs supplémentaires.

$$\frac{\Gamma \vdash expr_1 : (\tau_1 \text{ -> } \tau_2) \qquad \Gamma \vdash expr_2 : \tau_1}{\Gamma \vdash \{expr_1\ expr_2\} : \tau_2}$$

UNIVERSITÉ CÔTE D'AZUR

Sous-types

- Un type `τ` est un sous-type du type `τ'` (on note `τ ≤ τ'`) lorsqu'une expression du type `τ` peut être utilisée à la place d'une expression du type `τ'`.

```
{[x : num] [y : num]} ≤ {[x : num]}

{[y : num] [x : num]} ≤ {[x : num] [y : num]}}

{[x : num]} ≤ {[x : num]}

{[x : num]} ≰ {[x : num] [y : num]}
```

UNIVERSITÉ CÔTE D'AZUR

Règles pour le sous-typage

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i = \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

$$\texttt{num} \leq \texttt{num}$$

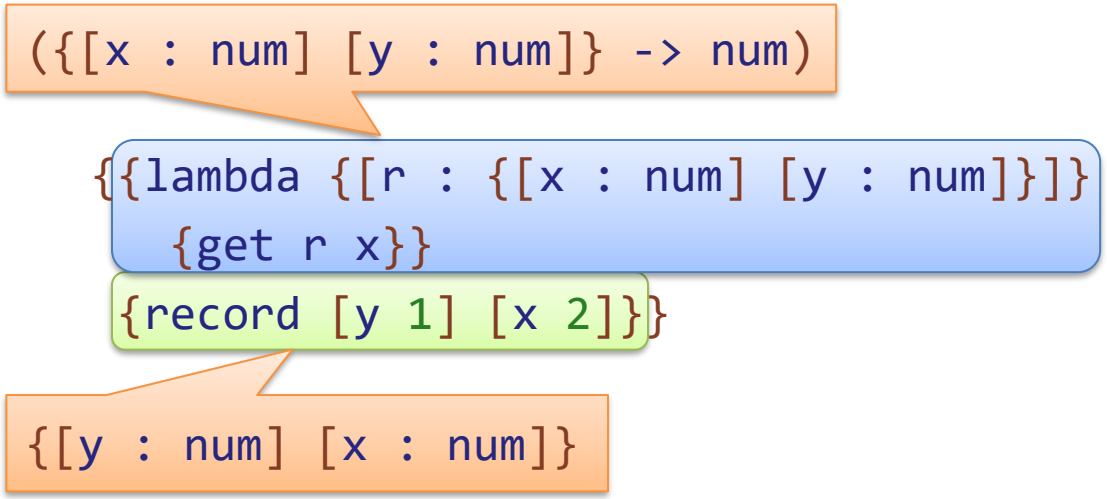
$$\texttt{bool} \leq \texttt{bool}$$

$$(\tau_1 \texttt{ -> } \tau_2) \leq (\tau_1 \texttt{ -> } \tau_2)$$

$$\frac{\Gamma \vdash expr_1 : (\tau_1 \texttt{ -> } \tau_2) \qquad \Gamma \vdash expr_2 : \tau_3 \qquad \tau_3 \leq \tau_1}{\Gamma \vdash \{expr_1\ expr_2\} : \tau_2}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs

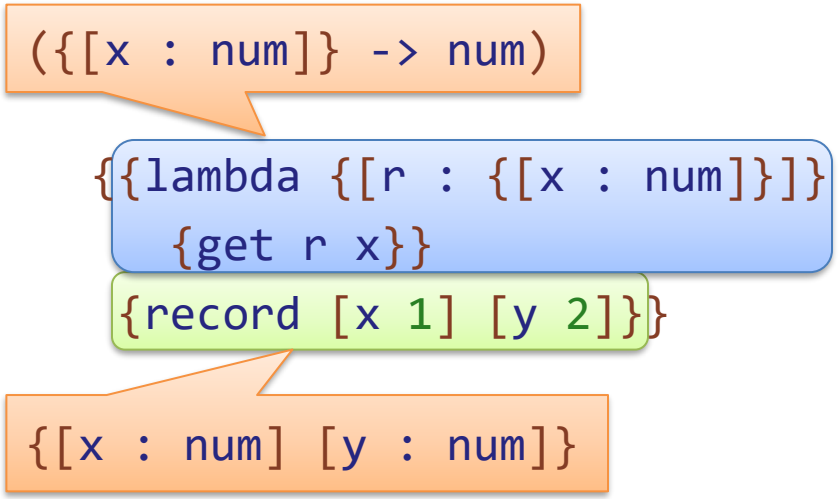


L'expression a pour type `num` car

$$\{[y : num]\ [x : num]\} \leq \{[x : num]\ [y : num]\}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs

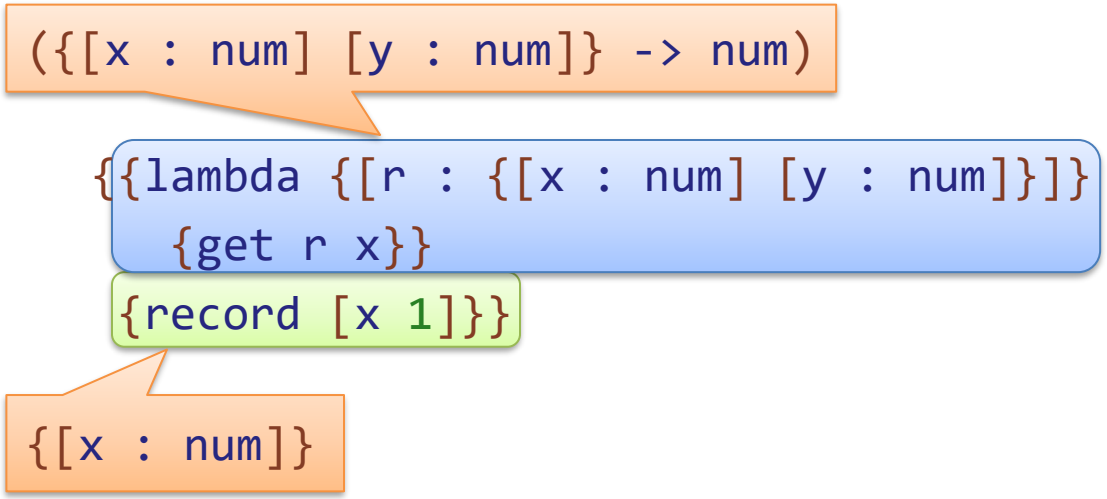


L'expression a pour type `num` car

$$\{[x : num]\ [y : num]\} \leq \{[x : num]\}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs



Non-typable car

$$\{[x : num]\} \not\leq \{[x : num]\ [y : num]\}$$

UNIVERSITÉ CÔTE D'AZUR

Typage, enregistrements et champs

```
({[p : {[x : num]}]} -> num)

{{lambda {[r : {[p : {[x : num]}]}]}
   {get {get r p} x}}
 {record [p {record [x 1] [y 2]}]}}

{[p : {[x : num] [y : num]}]}
```

Non-typable car

$$\{[p : \{[x : num]\ [y : num]\}]\} \not\leq \{[p : \{[x : num]\}]\}$$

Est-ce ce que l'on attend ?

UNIVERSITÉ CÔTE D'AZUR

Règles pour le sous-typage

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i \leq \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

$$\{[x : num]\ [y : num]\} \leq \{[x : num]\}$$

$$\Rightarrow$$

$$\{[p : \{[x : num]\ [y : num]\}]\} \leq \{[p : \{[x : num]\}]\}$$

Le sous-typage doit être récursif !

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs

```
{set {record {p {record {x 1}}}}
     p
     {record {x 1} {y 2}}}
```

- Non-typable avec la règle actuelle.

$$\frac{\Gamma \vdash expr_1 : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \qquad x = x_i \qquad \Gamma \vdash expr_2 : \tau_i}{\Gamma \vdash \{set\ expr_1\ x\ expr_2\} : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\}}$$

- Règle modifiée

$$\frac{\Gamma \vdash expr_1 : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \qquad x = x_i \qquad \Gamma \vdash expr_2 : \tau \qquad \tau \leq \tau_i}{\Gamma \vdash \{set\ expr_1\ x\ expr_2\} : \{[x_1 : \tau_1] \ldots [x_n : \tau_n]\}}$$

UNIVERSITÉ
CÔTE D'AZUR

SOUS-TYPAGE ET FONCTIONS

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemples

Je voudrais changer de portable !

Pourquoi pas un smartphone ?

`{[val : num]}`

`{[val : num] [gen : num]}`

OK

$\{[val : num]\ [gen : num]\} \leq \{[val : num]\}$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 1

Tu l'achèterais où ton portable ?

Je connais un bon magasin de smartphones.

```
(num -> {[val : num]})
```

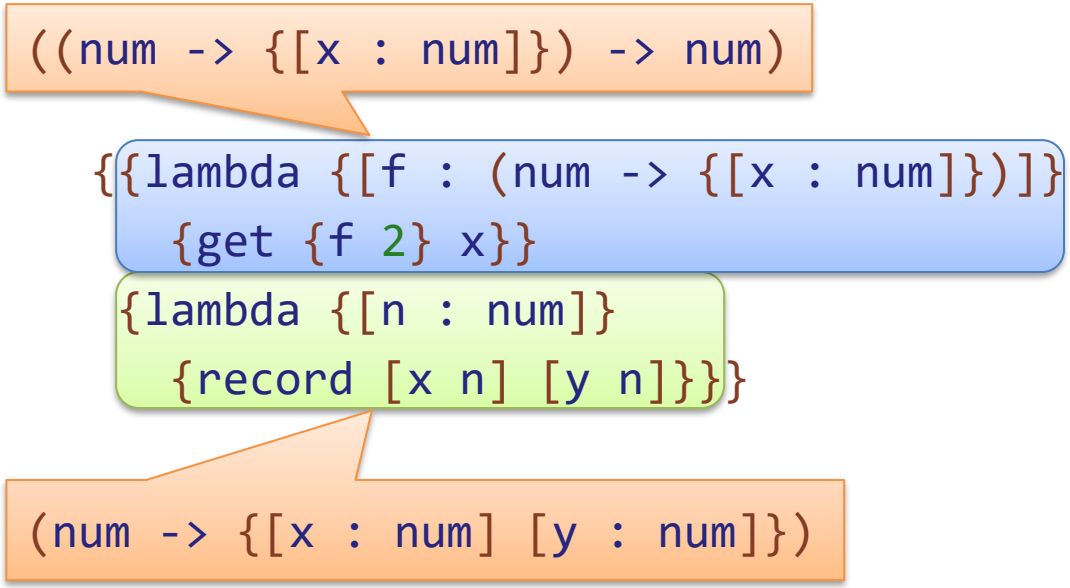
```
(num -> {[val : num] [gen : num]})
```

OK

$$(num \rightarrow \{[val : num]\ [gen : num]\}) \leq (num \rightarrow \{[val : num]\})$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 1



Non-typable avec les règles actuelles :

$$(\text{num} \to \{[x : \text{num}]\ [y : \text{num}]\}) \not\leq (\text{num} \to \{[x : \text{num}]\})$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 1

- Comparer

```
(num -> {[x : num] [y : num]})
```

et

```
(num -> {[x : num]})
```

- Règle actuelle

$$(\tau_1 \text{ -> } \tau_2) \leq (\tau_1 \text{ -> } \tau_2)$$

- Nouvelle règle pour autoriser la comparaison

$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \text{ -> } \tau_2) \leq (\tau_1 \text{ -> } \tau'_2)}$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 2

Ce magasin ne rachète que des smartphones.

Je veux revendre mon ancien portable.

```
({[val : num] [gen : num]} -> num)
```

```
({[val : num]} -> num)
```

NON

$$\texttt{({[val : num] [gen : num]} -> num)} \not\leq \texttt{({[val : num]} -> num)}$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 2

```
(({[x : num]} -> num) -> num)

{{lambda {[f : ({[x : num]} -> num)]}
   {f {record [x 1]}}}
 {lambda {[r : {[x : num] [y : num]}]}
   {get r y}}}

({[x : num] [y : num]} -> num)
```

Non-typable car

$$(\{[x : num]\ [y : num]\} \to num) \not\leq (\{[x : num]\} \to num)$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 2

- Comparer

```
({[x : num] [y : num]} -> num)
```

et

```
({[x : num]} -> num)
```

- Règle actuelle

$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \text{ -> } \tau_2) \leq (\tau_1 \text{ -> } \tau'_2)}$$

- La comparaison est correctement rejetée

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 3

Je connais un autre magasin qui reprend tout type de téléphone.

```
({[val : num]} -> num)
```

Finalement, le smartphone ne me convient pas…

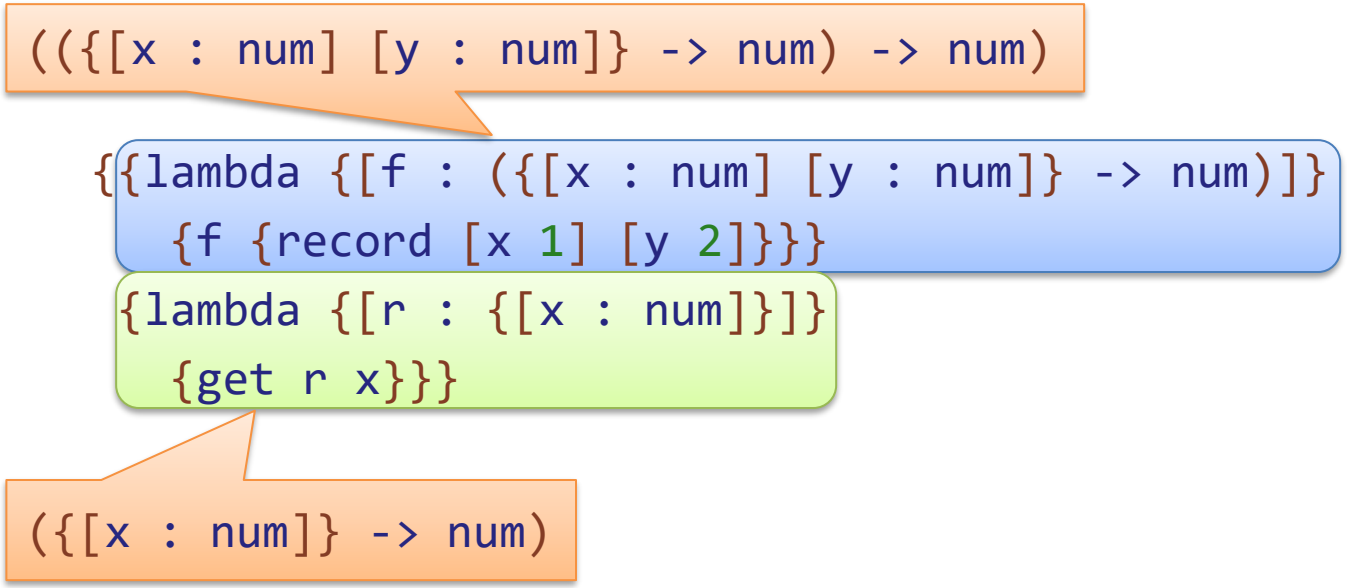
```
({[val : num] [gen : num]} -> num)
```

OK

$$\texttt{({[val : num]} -> num)} \leq \texttt{({[val : num] [gen : num]} -> num)}$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 3



Non-typable avec les règles actuelles :

$$(\{[x : num]\} \rightarrow num) \not\leq (\{[x : num]\ [y : num]\} \rightarrow num)$$

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et fonctions : Exemple 3

- Comparer

```
({[x : num]} -> num)
```

et

```
({[x : num] [y : num]} -> num)
```

- Règle actuelle

$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \text{ -> } \tau_2) \leq (\tau_1 \text{ -> } \tau'_2)}$$

- Nouvelle règle pour autoriser la comparaison

$$\frac{\tau'_1 \leq \tau_1 \qquad \tau_2 \leq \tau'_2}{(\tau_1 \text{ -> } \tau_2) \leq (\tau'_1 \text{ -> } \tau'_2)}$$

UNIVERSITÉ CÔTE D'AZUR

Covariance et contravariance

$$\frac{\tau'_1 \leq \tau_1 \qquad \tau_2 \leq \tau'_2}{(\tau_1 \text{ -> } \tau_2) \leq (\tau'_1 \text{ -> } \tau'_2)}$$

- On dit que le type de retour des fonctions est **covariant** avec le type des fonctions.

- On dit que le type des paramètres des fonctions est **contravariant** avec le type des fonctions.

UNIVERSITÉ CÔTE D'AZUR

Covariance et contravariance

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i \leq \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

- Le type des champs est covariant avec le type des enregistrements

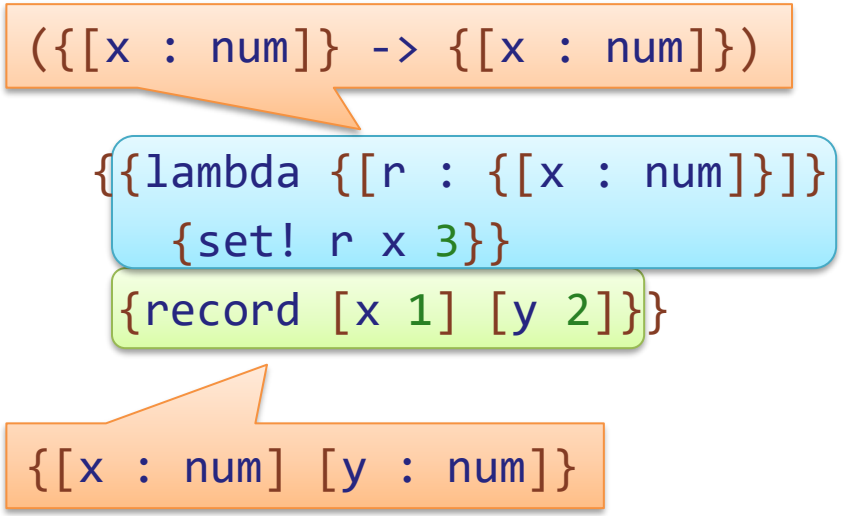
… du moins tant que les mises à jour sont fonctionnelles.

UNIVERSITÉ
CÔTE D'AZUR

SOUS-TYPAGE ET MUTATIONS

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs



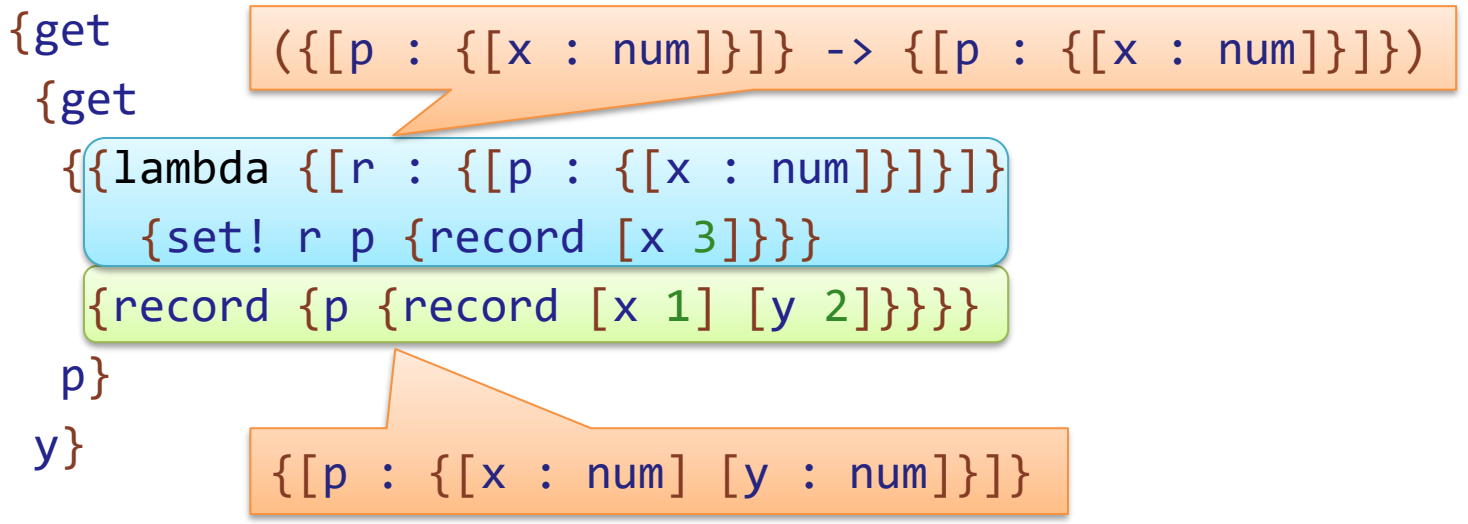
$$\frac{\Gamma \vdash expr_1 : (\tau_1 \text{ -> } \tau_2) \qquad \Gamma \vdash expr_2 : \tau_3 \qquad \tau_3 \leq \tau_1}{\Gamma \vdash \{expr_1\ expr_2\} : \tau_2}$$

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i \leq \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

Les règles semblent correctes.

UNIVERSITÉ CÔTE D'AZUR

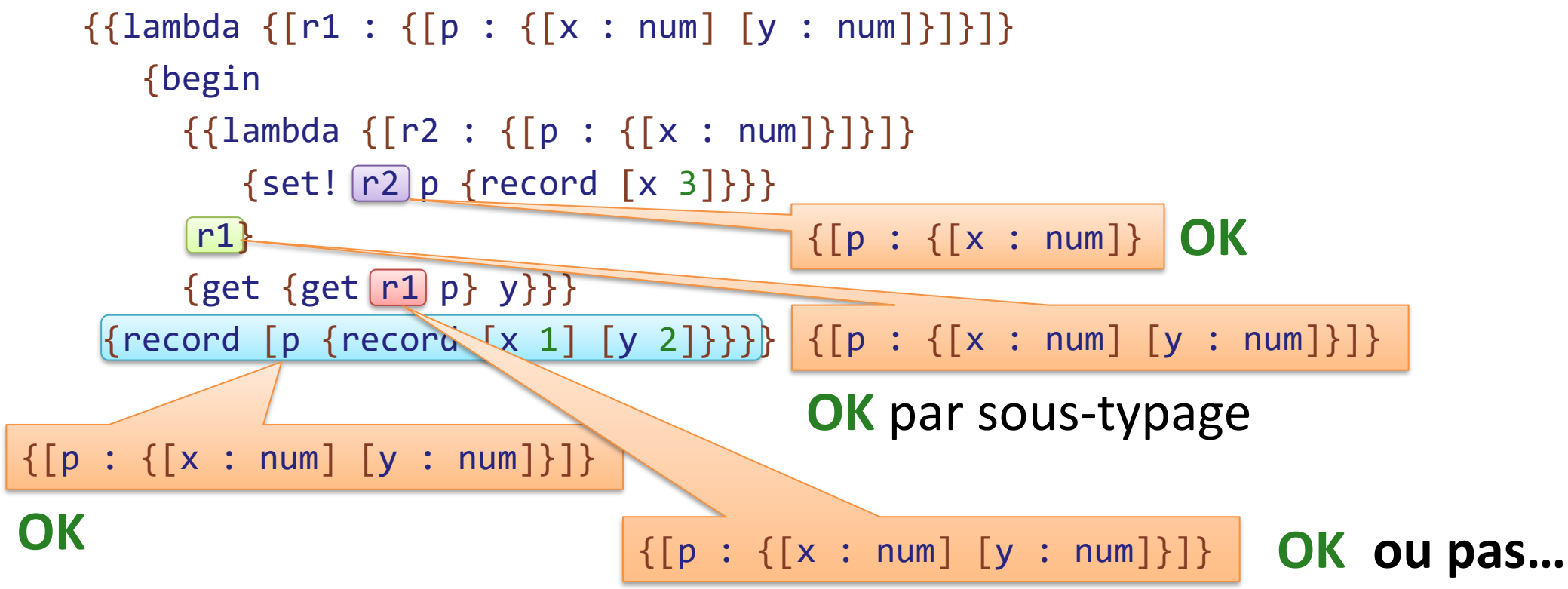
Sous-typage et mise à jour des champs



Non-typable

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs



UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs

```
{{lambda {[r1 : {[p : {[x : num] [y : num]}]}]}
   {begin
     {{lambda {[r2 : {[p : {[x : num]}]}]}
        {set! r2 p {record [x 3]}}}
      r1}
     {get {get r1 p} y}}}
 {record [p {record [x 1] [y 2]}]}}
```

- L'expression est valide selon le système de typage actuel, mais que se passe-t-il si on tente de l'évaluer ?

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs

- Comparer

$$\{[p : \{[x : num]\ [y : num]\}]\}$$

et

$$\{[p : \{[x : num]\}]\}$$

- Règle actuelle

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i \leq \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

- Ne fonctionne pas pour la mise à jour impérative des champs.

UNIVERSITÉ CÔTE D'AZUR

Sous-typage et mise à jour des champs

- Comparer

```
{[p : {[x : num] [y : num]}]}
```

et

```
{[p : {[x : num]}]}
```

- Nouvelle règle

$$\frac{\{x_1, \ldots, x_n\} \supseteq \{x'_1, \ldots, x'_m\} \qquad x_i = x'_j \Rightarrow \tau_i = \tau'_j}{\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}}$$

- Avec la mise à jour impérative, le type des champs doit être **invariant**.