Part I

Typed Records

```
<Expr>  ::=  <Num>
          |  {+ <Expr> <Expr>}
          |  {* <Expr> <Expr>}
          |  <Sym>
          |  {lambda {[<Sym> : <Type>]} <Expr>}
          |  {<Expr> <Expr>}
          |  {record {<Sym> <Expr>} ...}
          |  {get <Expr> <Sym>}
          |  {set <Expr> <Sym> <Expr>}

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

NEW NEW NEW NEW

Records

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

Has type

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

Records

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

Records

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

Records

```
{set {record {x {+ 1 2}}
             {y {* 3 4}}}
     y
     0}
```

Same value as `{record {x 3}`
`{y 0}}`

Records

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

Records

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

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

Records

```
{lambda {[r : {[x : num]}]}
  {set r x 1}}
```

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

Records

```
{let {[f : ({[x : num]} -> {[x : num]})
          {lambda {[r : {[x : num]}]}
            {set r x 1}}]}
  {f {record {x {+ 1 1}}}}}
```

Typechecking

$$\frac{\Gamma \vdash \mathbf{e}_1 : \tau_1 \quad \ldots \quad \Gamma \vdash \mathbf{e}_n : \tau_n \qquad \tau = \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\}}{\Gamma \vdash \{\mathtt{record}\ \{\{\mathbf{x}_1 : \mathbf{e}_1\} \ \ldots \ \{\mathbf{x}_n : \mathbf{e}_n\}\}\} : \tau}$$

$$\frac{\Gamma \vdash \mathbf{e} : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\} \qquad \mathbf{x}_i \in \{\mathbf{x}_1, \ \ldots \ \mathbf{x}_n\}}{\Gamma \vdash \{\mathtt{get}\ \mathbf{e}\ \mathbf{x}_i\} : \tau_i}$$

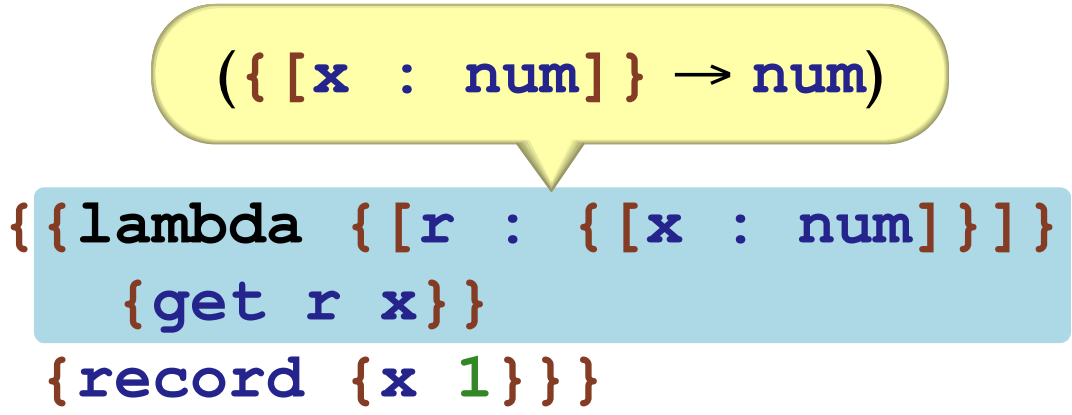
$$\frac{\Gamma \vdash \mathbf{e}_1 : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\} \quad \Gamma \vdash \mathbf{e}_2 : \tau_i \qquad \mathbf{x}_i \in \{\mathbf{x}_1, \ \ldots \ \mathbf{x}_n\}}{\Gamma \vdash \{\mathtt{set}\ \mathbf{e}_1\ \mathbf{x}_i\ \mathbf{e}_2\} : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\}}$$

Part 2

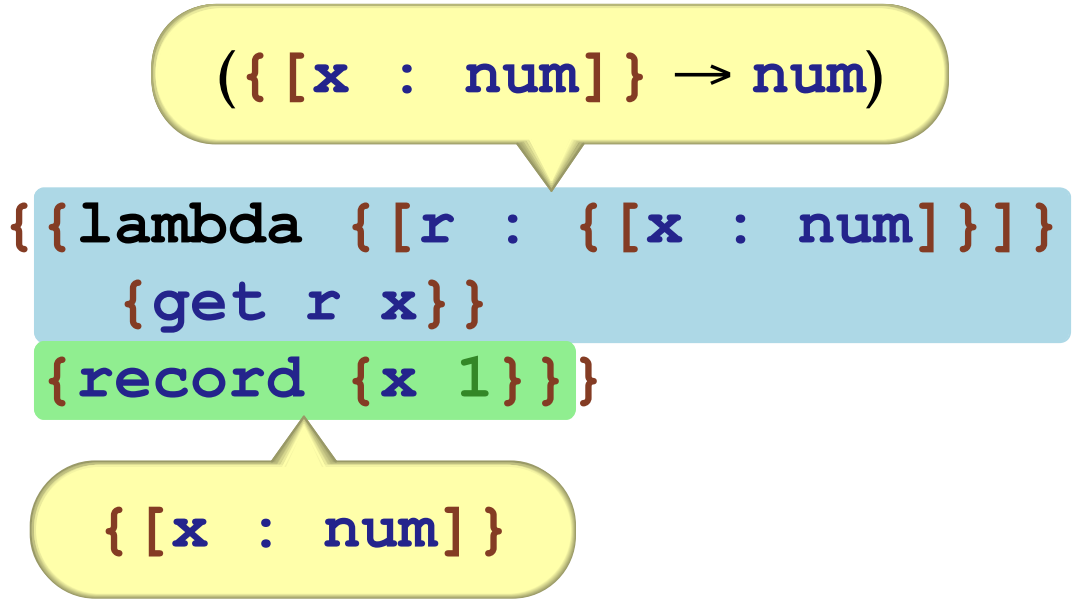
Records and Fields

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

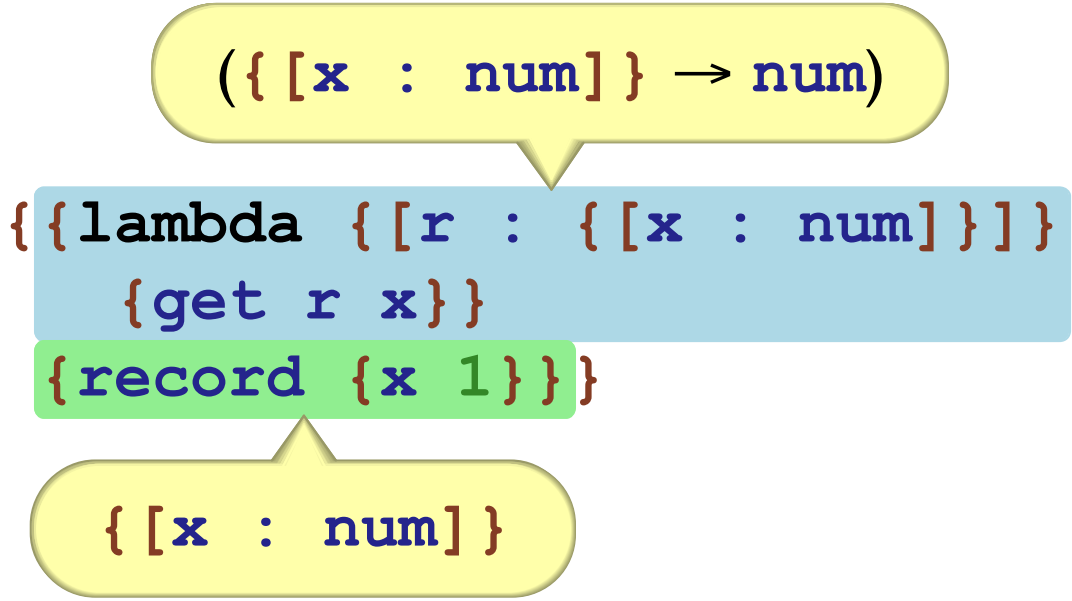
Records and Fields



Records and Fields



Records and Fields



Has type **num**

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \qquad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1\ \mathbf{e}_2\} : \tau_3}$$

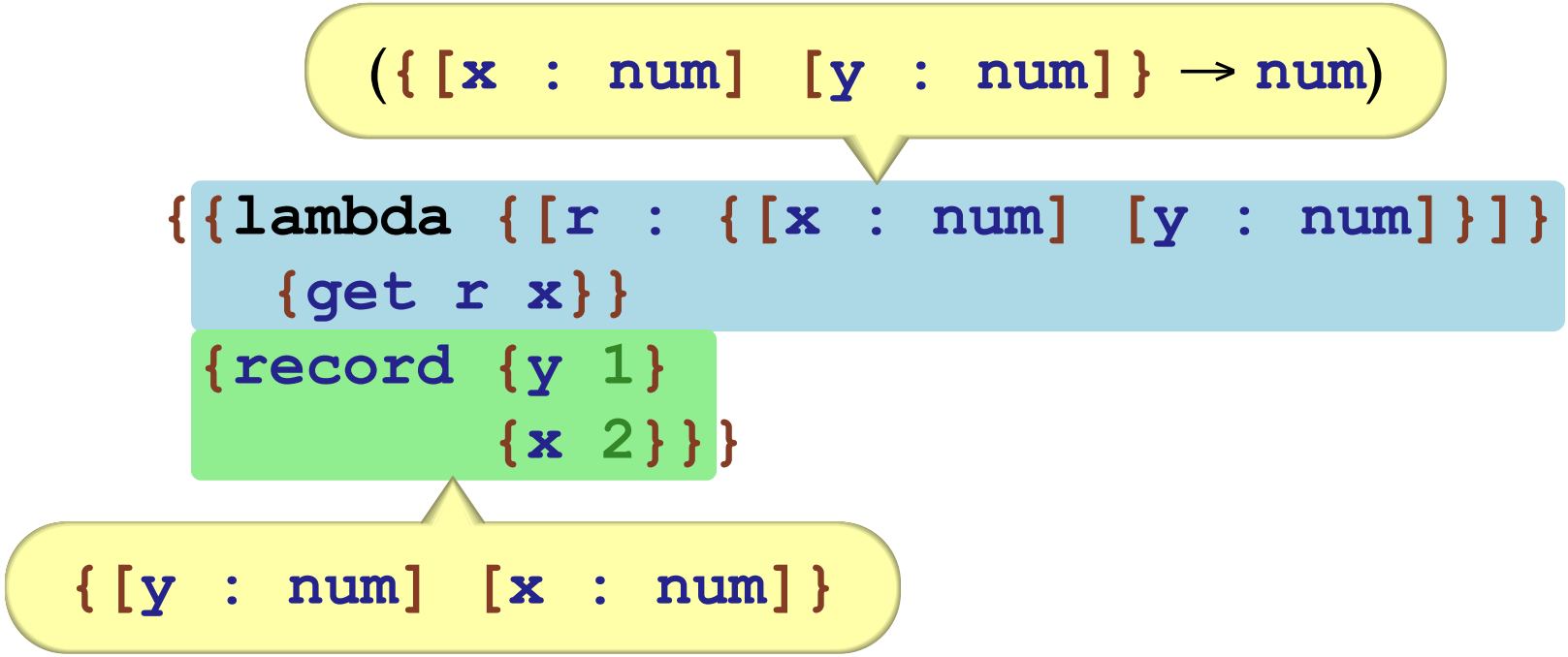
Records and Fields

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

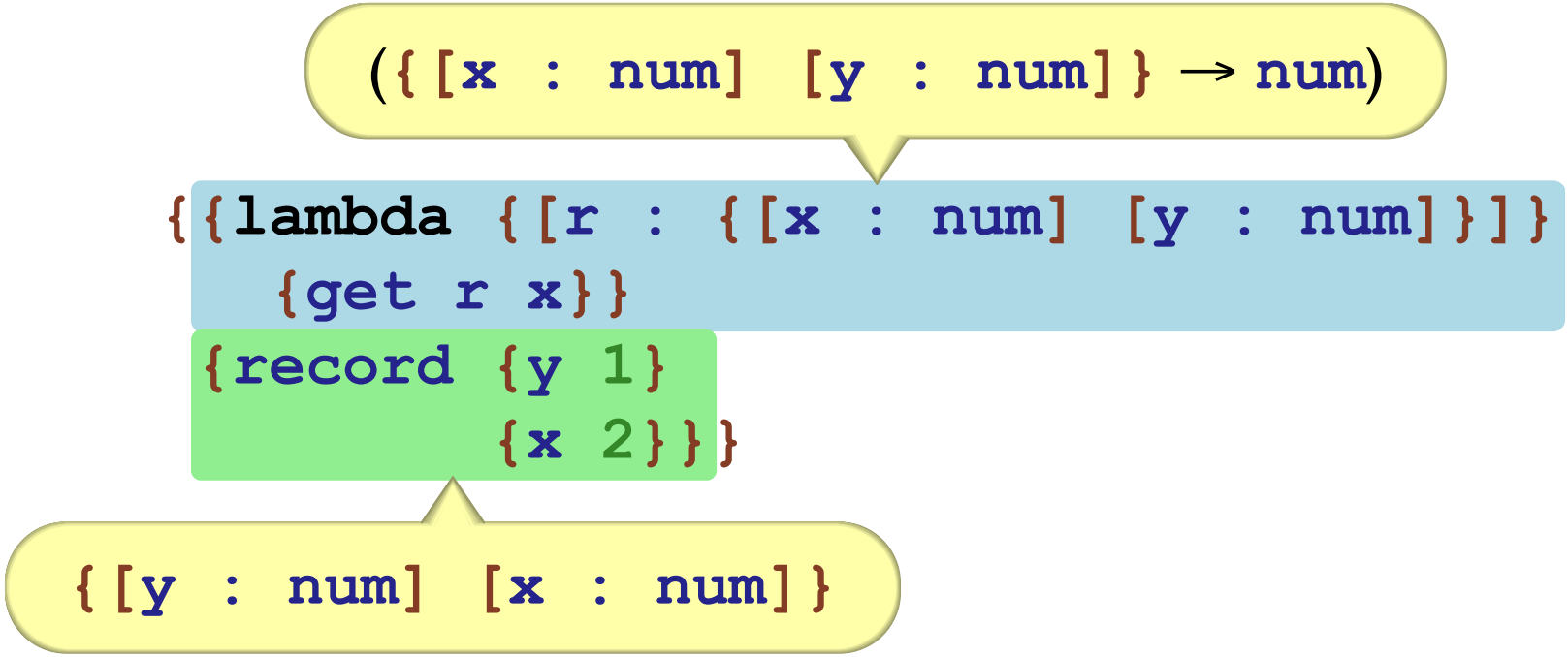
Records and Fields

```
({[x : num] [y : num]} → num)
{{lambda {[r : {[x : num] [y : num]}]}
   {get r x}}
 {record {y 1}
         {x 2}}}
```

Records and Fields



Records and Fields



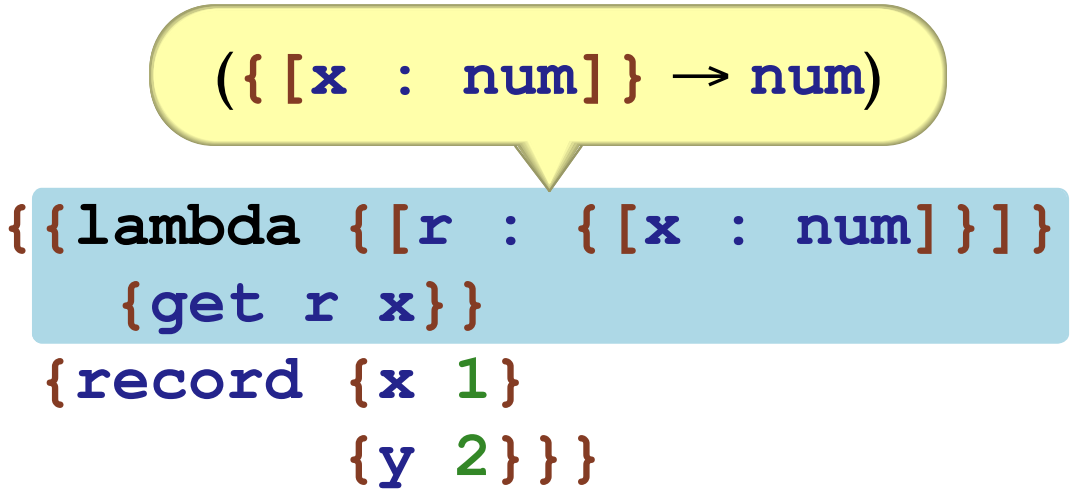
no type — field order doesn't match

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \qquad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1\ \mathbf{e}_2\} : \tau_3}$$

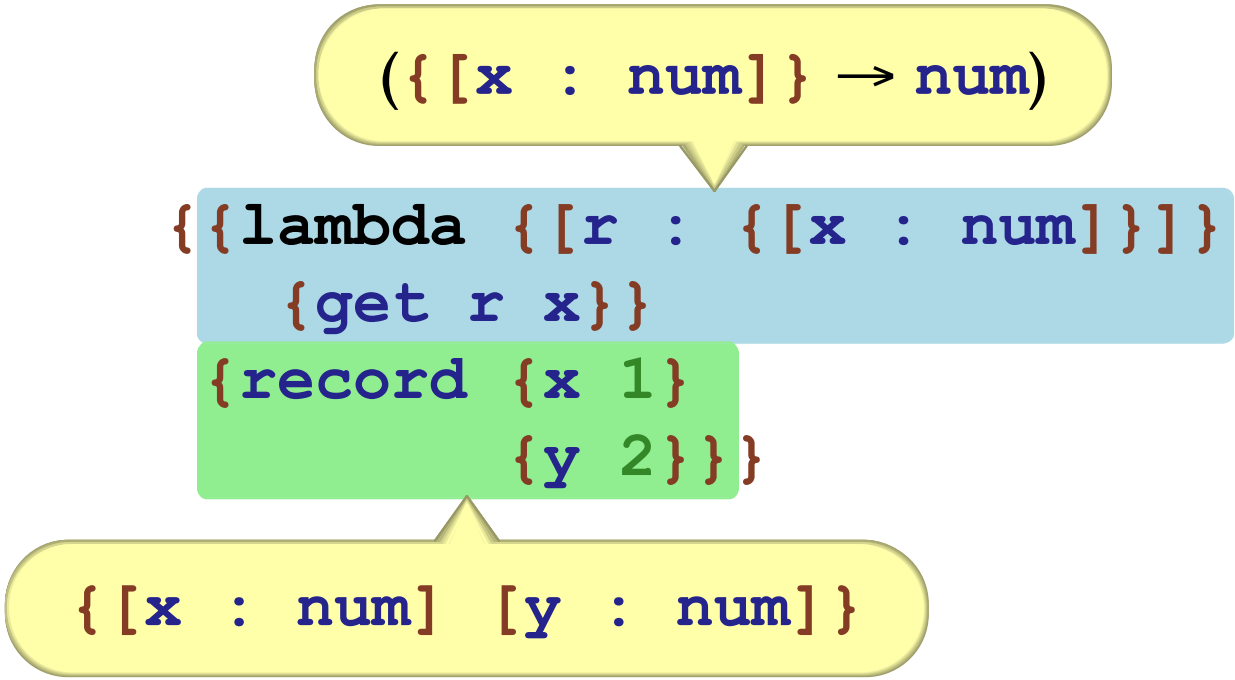
Records and Fields

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

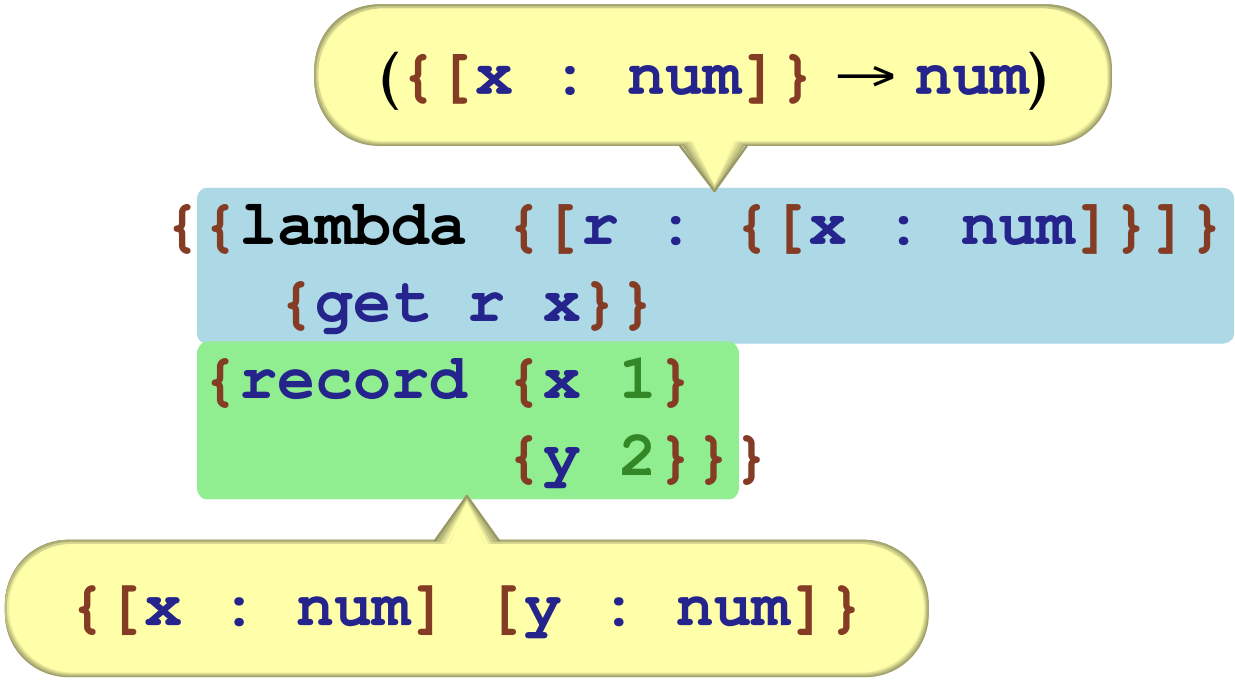
Records and Fields



Records and Fields



Records and Fields



no type — extra fields in argument

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_2 \rightarrow \tau_3) \qquad \Gamma \vdash \mathbf{e}_2 : \tau_2}{\Gamma \vdash \{\mathbf{e}_1\ \mathbf{e}_2\} : \tau_3}$$

Subtypes

If τ is a ***subtype*** of τ',

$$\tau \leq \tau'$$

then an expression of type τ can be used in place of an expression of type τ'

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

Subtypes

If τ is a ***subtype*** of τ',

$$\tau \leq \tau'$$

then an expression of type τ can be used in place of an expression of type τ'

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

Subtypes

If τ is a ***subtype*** of τ',

$$\tau \leq \tau'$$

then an expression of type τ can be used in place of an expression of type τ'

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

Subtypes

If τ is a ***subtype*** of τ',

$$\tau \leq \tau'$$

then an expression of type τ can be used in place of an expression of type τ'

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

Subtypes

If τ is a ***subtype*** of τ',

$$\tau \leq \tau'$$

then an expression of type τ can be used in place of an expression of type τ'

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

Intution: $\tau \leq \tau'$ means that fewer values fit τ than τ'

Subtype Rules

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

$$\texttt{num} \leq \texttt{num} \qquad \texttt{bool} \leq \texttt{bool}$$

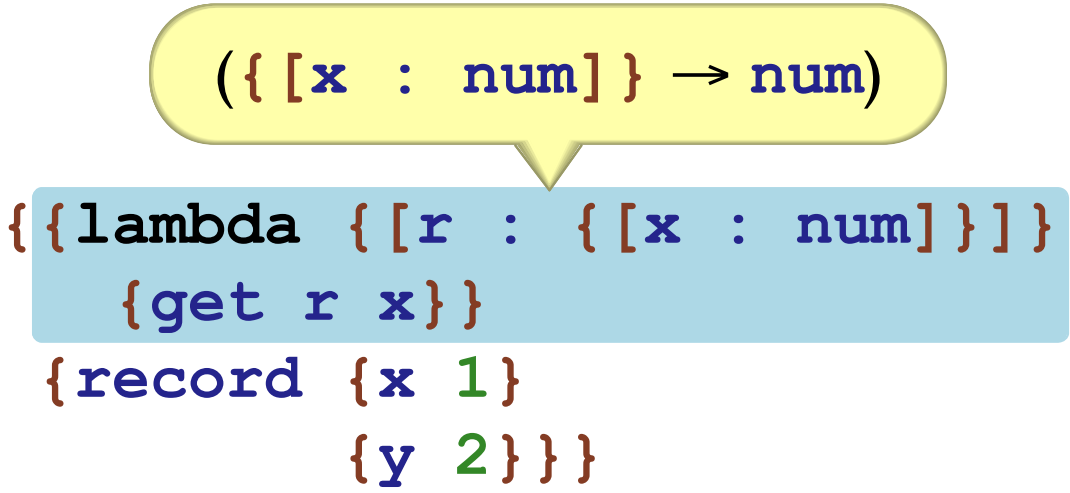
$$(\tau_1 \rightarrow \tau_2) \leq (\tau_1 \rightarrow \tau_2)$$

$$\frac{\Gamma \vdash \mathbf{e}_1 : (\tau_1 \rightarrow \tau_3) \qquad \Gamma \vdash \mathbf{e}_2 : \tau_2 \qquad \tau_2 \leq \tau_1}{\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3}$$

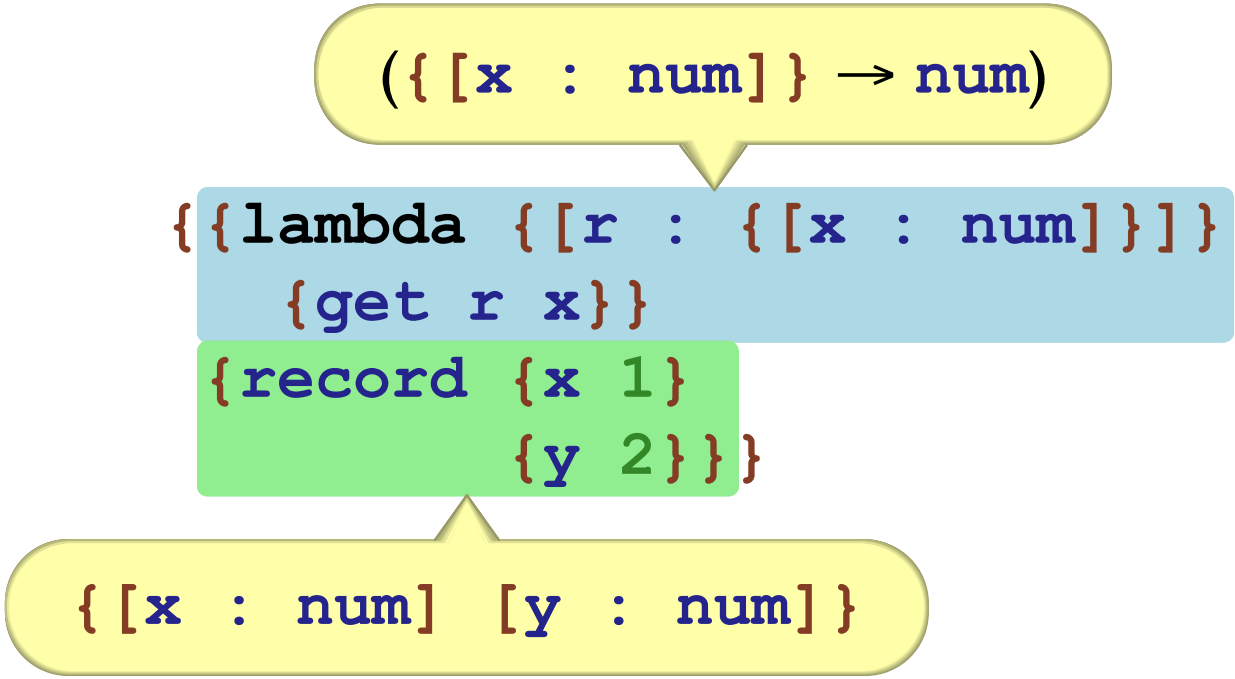
Records and Fields

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

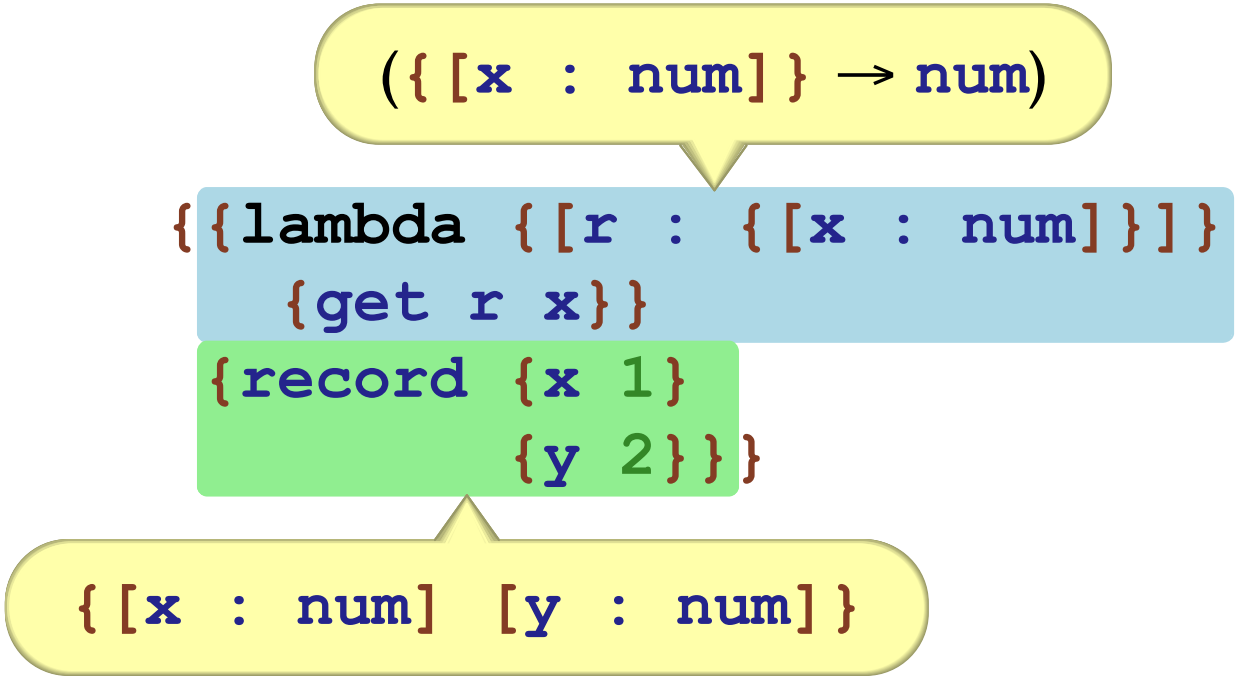
Records and Fields



Records and Fields



Records and Fields



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

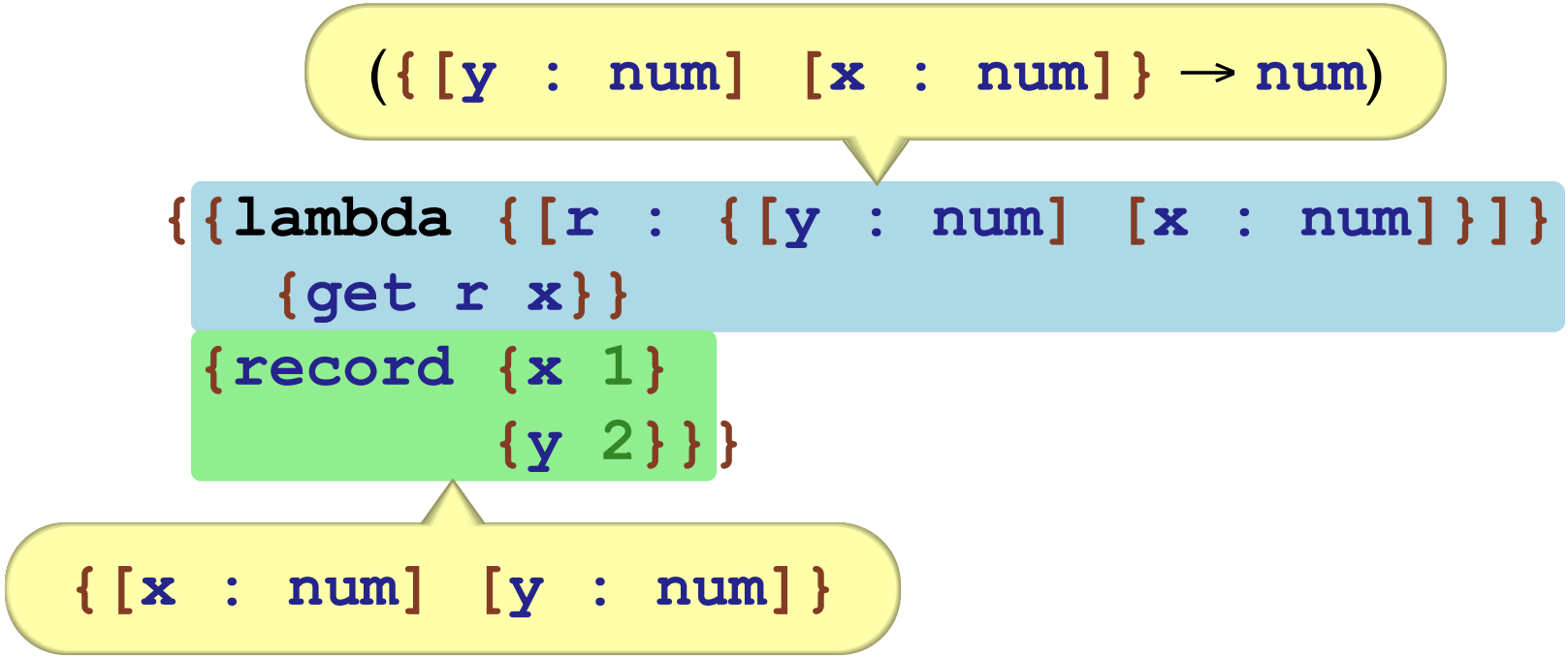
Records and Fields

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

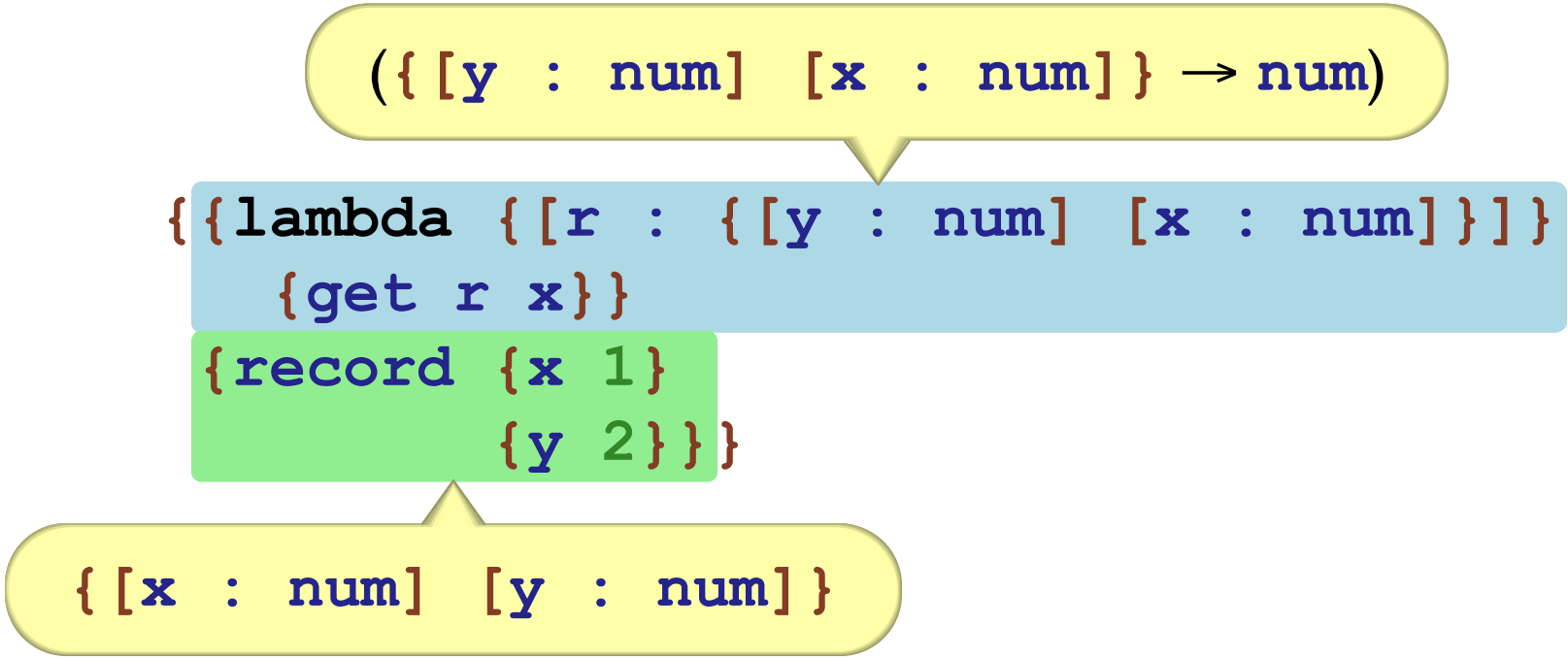
Records and Fields

```
({[y : num] [x : num]} → num)
{{lambda {[r : {[y : num] [x : num]}]}
   {get r x}}
 {record {x 1}
         {y 2}}}
```

Records and Fields



Records and Fields

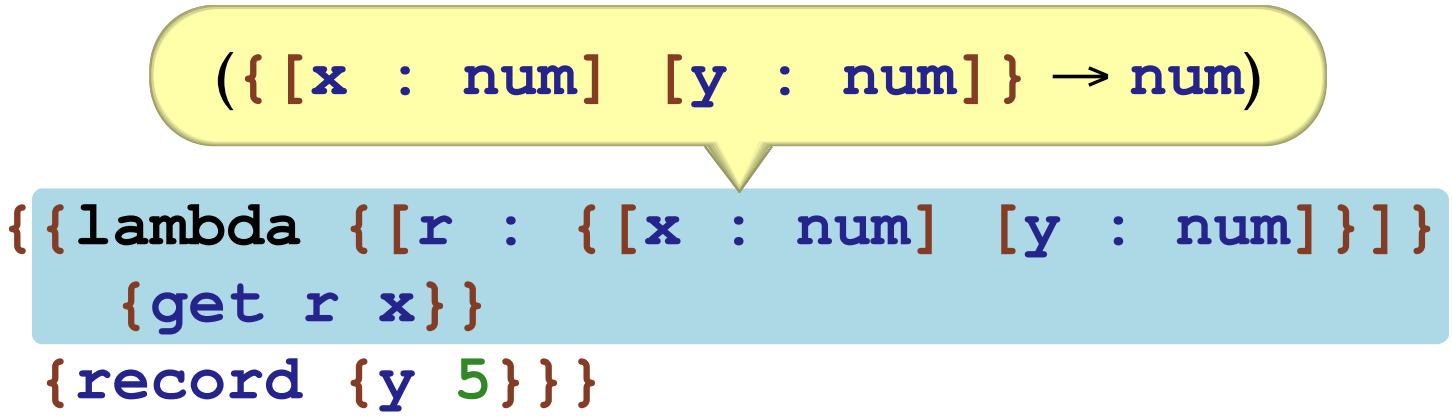


$\{[x : num]\ [y : num]\} \leq \{[y : num]\ [x : num]\}$

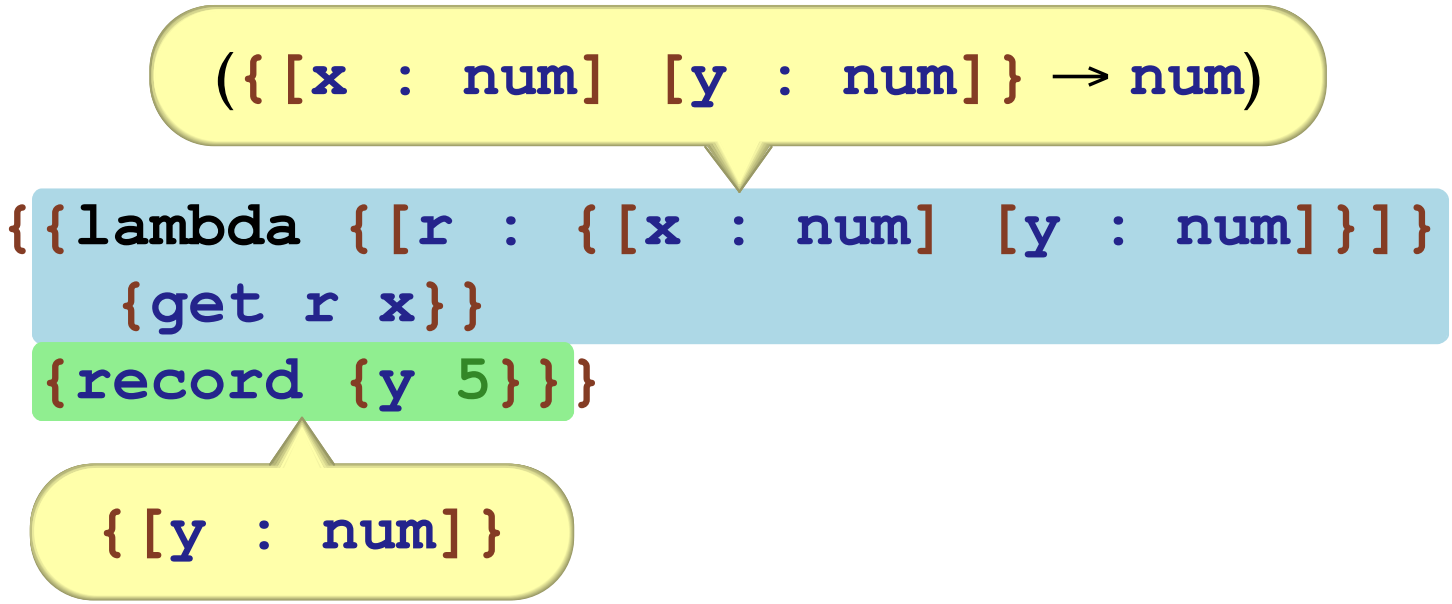
Records and Fields

```
{{lambda {[r : {[x : num] [y : num]}]}
   {get r x}}
 {record {y 5}}}
```

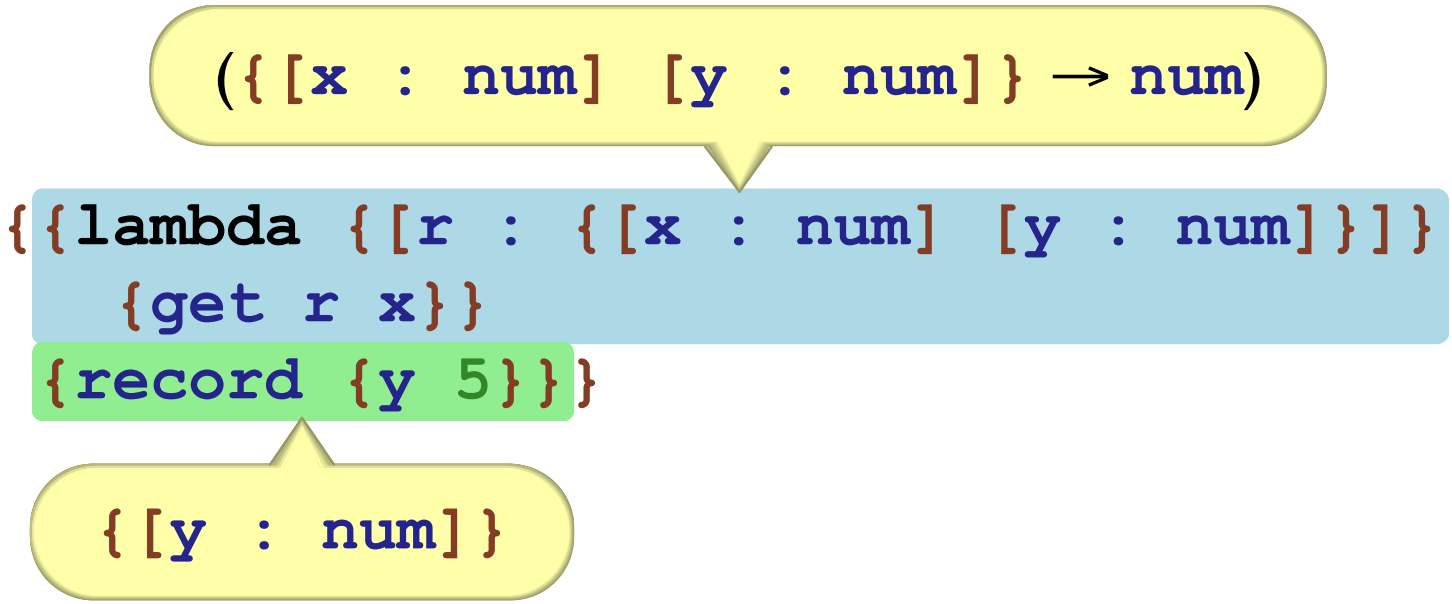
Records and Fields



Records and Fields



Records and Fields



{[y : num]} ≰ {[x : num] [y : num]}

Part 3

Subtypes in Fields

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {get {get r p} x}}]}
  {f {record {p {record {x 5}
                        {y 6}}}}}}
```

Subtypes in Fields

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {get {get r p} x}}]}
  {f {record {p {record {x 5}
                        {y 6}}}}}}
```

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

Subtypes in Fields

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {get {get r p} x}}]}
  {f {record {p {record {x 5}
                        {y 6}}}}}}
```

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

vs.

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

Subtypes in Fields

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

vs.

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

$$\{\mathbf{x}_1, \ldots \mathbf{x}_n\} \supseteq \{\mathbf{x}'_1, \ldots \mathbf{x}'_m\}$$

$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i = \tau'_j$$

$$\{[\mathbf{x}_1 : \tau_1] \ldots [\mathbf{x}_n : \tau_n]\} \leq \{[\mathbf{x}'_1 : \tau'_1] \ldots [\mathbf{x}'_m : \tau'_m]\}$$

Subtypes in Fields

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

vs.

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

$$\{\mathbf{x}_1, \ldots \mathbf{x}_n\} \supseteq \{\mathbf{x}'_1, \ldots \mathbf{x}'_m\}$$

$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i \leq \tau'_j$$

$$\{[\mathbf{x}_1 : \tau_1] \ldots [\mathbf{x}_n : \tau_n]\} \leq \{[\mathbf{x}'_1 : \tau'_1] \ldots [\mathbf{x}'_m : \tau'_m]\}$$

Field Update and Subtypes

```
{lambda {[r : {[p : {[x : num]}]}]}
  {set r p {record {x 5}
                   {y 6}}}}
```

Field Update and Subtypes

Original rule:

$$\frac{\Gamma \vdash \mathbf{e}_1 : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\} \quad \Gamma \vdash \mathbf{e}_2 : \tau_i \quad \mathbf{x}_i \in \{\mathbf{x}_1, \ \ldots \ \mathbf{x}_n\}}{\Gamma \vdash \{\mathtt{set}\ \mathbf{e}_1\ \mathbf{x}_i\ \mathbf{e}_2\} : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\}}$$

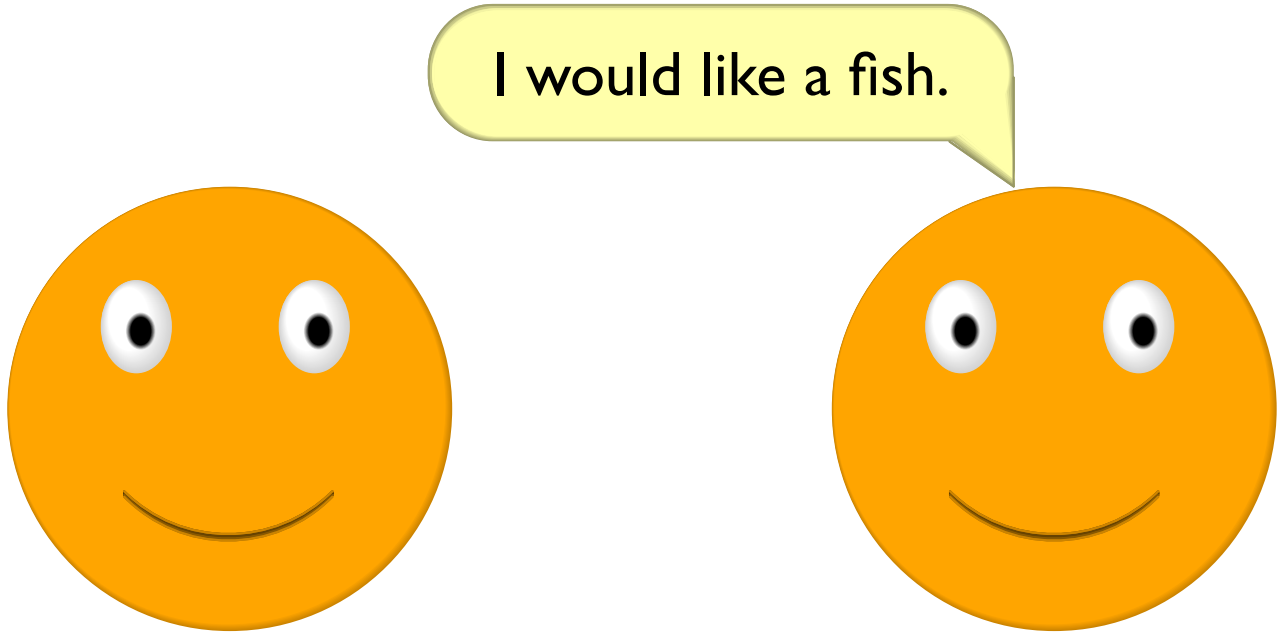
Revised rule:

$$\frac{\Gamma \vdash \mathbf{e}_1 : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\} \quad \Gamma \vdash \mathbf{e}_2 : \tau' \quad \mathbf{x}_i \in \{\mathbf{x}_1, \ \ldots \ \mathbf{x}_n\} \quad \tau' \leq \tau_i}{\Gamma \vdash \{\mathtt{set}\ \mathbf{e}_1\ \mathbf{x}_i\ \mathbf{e}_2\} : \{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\}}$$

Part 4

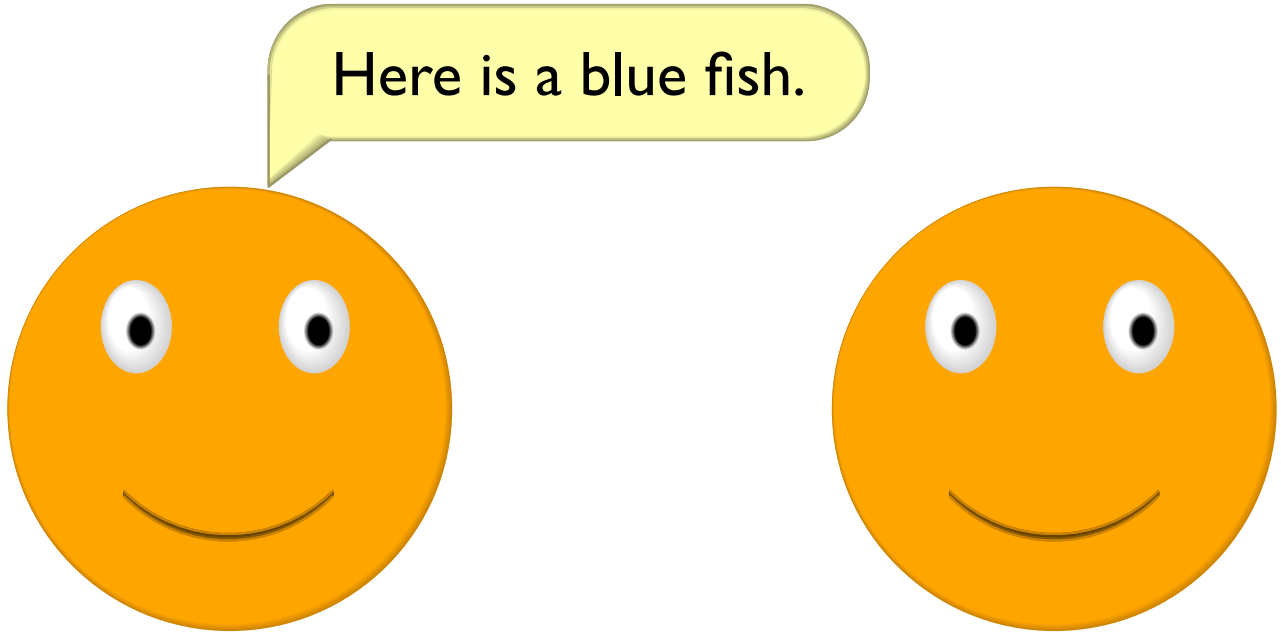
Subtypes and Functions

Subtypes and Functions




```
{[sz : num]}
```

Subtypes and Functions




```
{[sz : num]}
```

```
{[sz : num] [col : num]}
```

Subtypes and Functions

`{[sz : num]}`

`{[sz : num] [col : num]}`

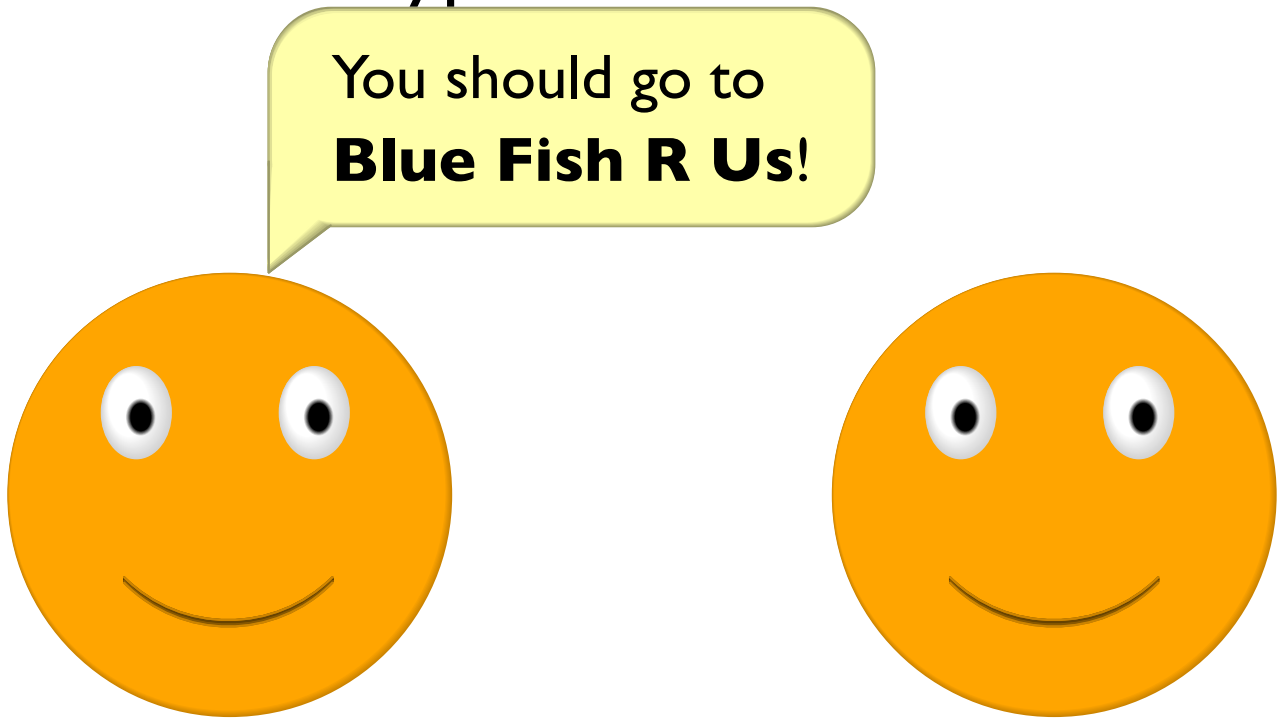
`{[sz : num] [col : num]}` ≤ `{[sz : num]}`

Subtypes and Functions

Subtypes and Functions

`(num → {[sz : num]})`

Subtypes and Functions



`(num → {[sz : num]})`

`(num → {[sz : num] [col : num]})`

Subtypes and Functions

(num → {[sz : num]})

(num → {[sz : num] [col : num]})

(num → {[sz : num] [col : num]}) ≤ (num → {[sz : num]})

Subtypes from Functions

```
{let {[f : ?
          {lambda {[g (num -> {[x : num]})]}
            {get {g 10} x}}]}
  {f {lambda {[v : num]}
       {record {x v}
               {y v}}}}}
```

Subtypes from Functions

```
{let {[f : ?
          {lambda {[g (num -> {[x : num]})]}
            {get {g 10} x}}]}
  {f {lambda {[v : num]}
       {record {x v}
               {y v}}}}}
```

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

Subtypes from Functions

```
{let {[f : ?
          {lambda {[g (num -> {[x : num]})]}
            {get {g 10} x}}]}
  {f {lambda {[v : num]}
       {record {x v}
               {y v}}}}}
```

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

vs.

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

Subtypes from Functions

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

vs.

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

$$(\tau_1 \rightarrow \tau_2) \leq (\tau_1 \rightarrow \tau_2)$$

Subtypes from Functions

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

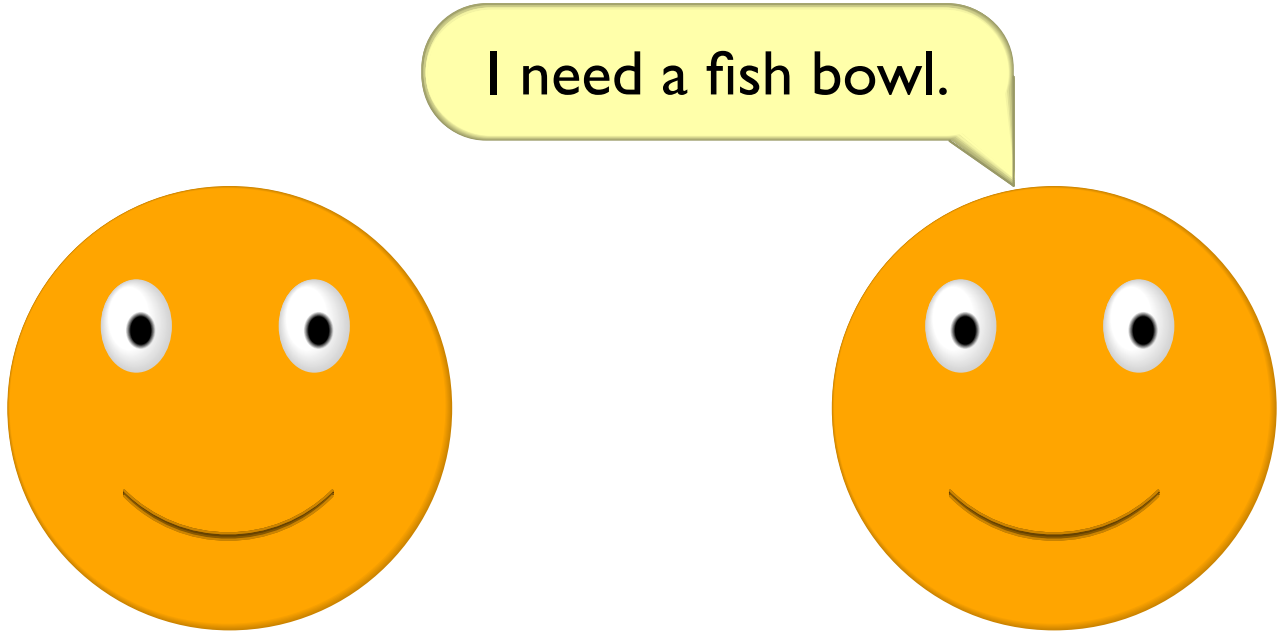
vs.

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

$$\frac{\tau_2 \leq \tau_2'}{(\tau_1 \to \tau_2) \leq (\tau_1 \to \tau_2')}$$

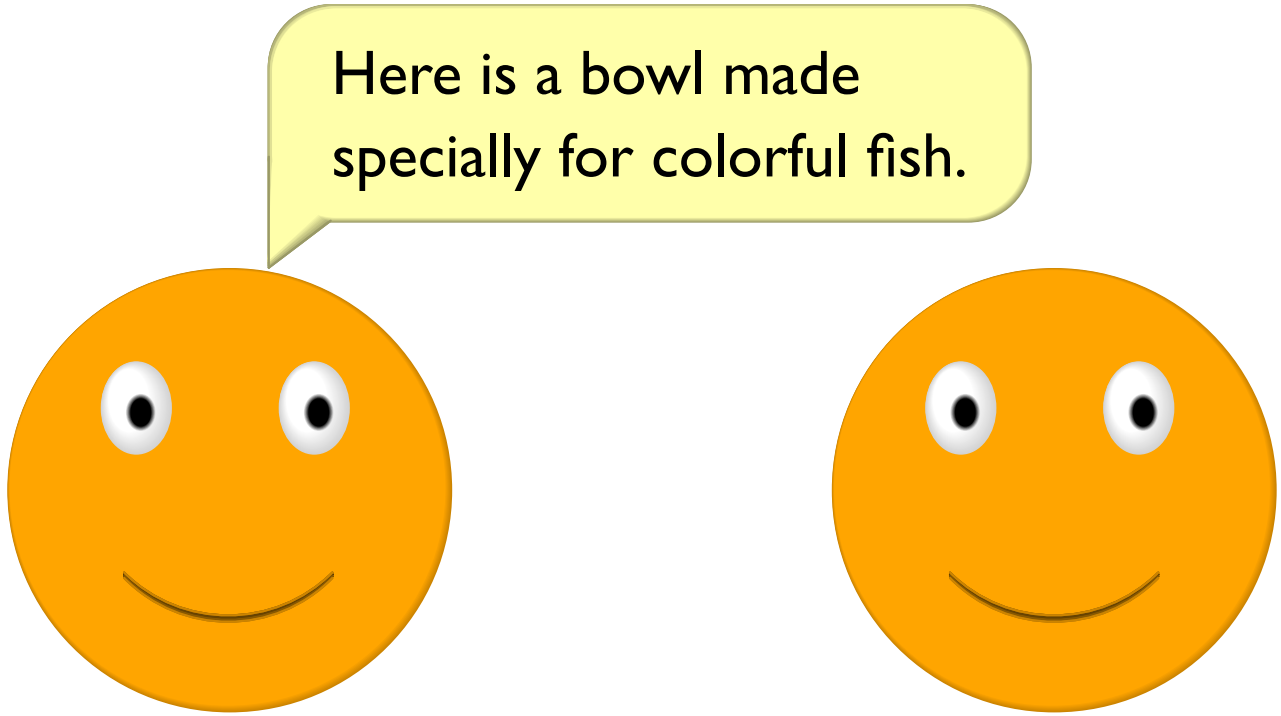
Subtype and Function Arguments

Subtype and Function Arguments



`({[sz : num]} → num)`

Subtype and Function Arguments



`({[sz : num]} → num)`

`({[sz : num] [col : num]} → num)`

Subtype and Function Arguments

$(\{[\texttt{sz} : \texttt{num}]\} \rightarrow \texttt{num})$

$(\{[\texttt{sz} : \texttt{num}]\ [\texttt{col} : \texttt{num}]\} \rightarrow \texttt{num})$

$(\{[\texttt{sz} : \texttt{num}]\ [\texttt{col} : \texttt{num}]\} \rightarrow \texttt{num}) \not\leq (\{[\texttt{sz} : \texttt{num}]\} \rightarrow \texttt{num})$

Subtypes and Function Arguments

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

Subtypes and Function Arguments

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

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

Subtypes and Function Arguments

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

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

vs.

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

Subtypes and Function Arguments

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

vs.

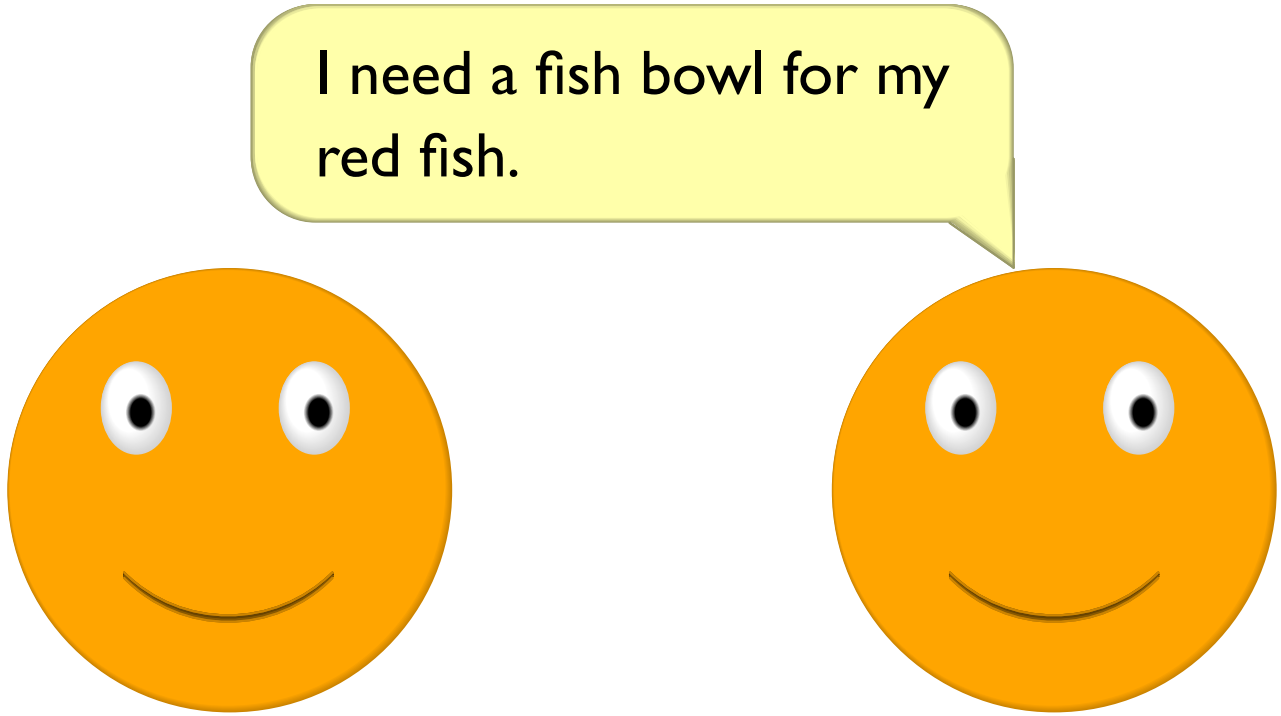
```
({[x : num]} -> num)
```

$$\frac{\tau_2 \leq \tau_2'}{(\tau_1 \to \tau_2) \leq (\tau_1 \to \tau_2')}$$

Correctly rejected!

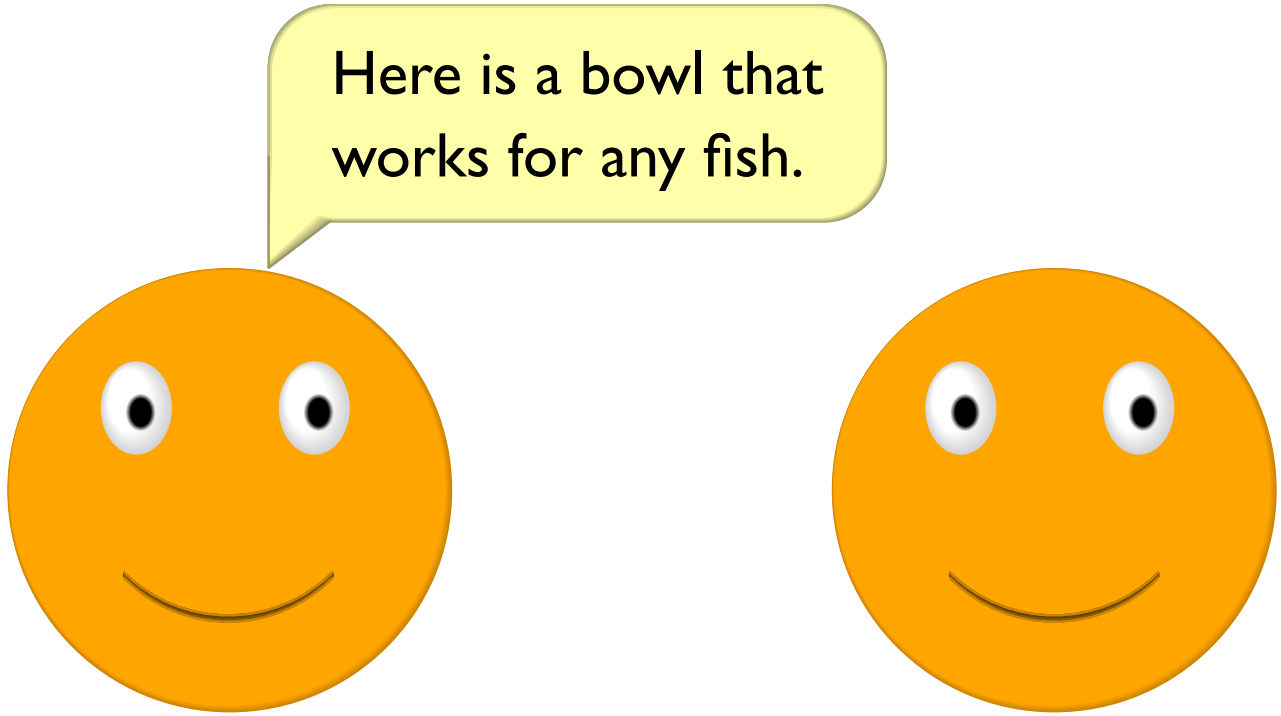
Subtype and Function Arguments

Subtype and Function Arguments




```
({[sz : num] [col : num]} → num)
```

Subtype and Function Arguments



({[sz : num] [col : num]} → num)

({[sz : num]} → num)

Subtype and Function Arguments

({[sz : num] [col : num]} → num)

({[sz : num]} → num)

({[sz : num]} → num) ≤ ({[sz : num] [col : num]} → num)

Subtypes to Functions

```
{let {[f : ?
          {lambda {[g ({[x : num] [y : num]} -> num)]}
            {g {record {x 1}
                       {y 2}}}}]}
  {f {lambda {[r : {[x : num]}]}
       {get r x}}}}
```

Subtypes to Functions

```
{let {[f : ?
          {lambda {[g ({[x : num] [y : num]} -> num)]}
            {g {record {x 1}
                       {y 2}}}}]}
  {f {lambda {[r : {[x : num]}]}
       {get r x}}}}
```

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

Subtypes to Functions

```
{let {[f : ?
          {lambda {[g ({[x : num] [y : num]} -> num)]}
            {g {record {x 1}
                       {y 2}}}}]}
  {f {lambda {[r : {[x : num]}]}
       {get r x}}}}
```

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

vs.

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

Subtypes to Functions

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

vs.

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

$$\frac{\tau_2 \leq \tau_2'}{(\tau_1 \to \tau_2) \leq (\tau_1 \to \tau_2')}$$

Subtypes to Functions

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

vs.

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

$$\frac{\tau_1' \leq \tau_1 \qquad \tau_2 \leq \tau_2'}{(\tau_1 \to \tau_2) \leq (\tau_1' \to \tau_2')}$$

Covariance and Contravariance

$$\frac{\tau_1' \leq \tau_1 \qquad \tau_2 \leq \tau_2'}{(\tau_1 \to \tau_2) \leq (\tau_1' \to \tau_2')}$$

Function-result types are ***covariant*** with function types

Function-argument types are ***contravariant*** with function types

Covariance and Contravariance

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

Field types are ***covariant*** with record types

... as long as `set` is a functional update

Part 5

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[x : num]}]}
            {set r x 5}}]}
  {f {record {x 1} {y 2}}}}
```

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[x : num]}]}
            {set r x 5}}]}
  {f {record {x 1} {y 2}}}}
```

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

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[x : num]}]}
            {set r x 5}}]}
  {f {record {x 1} {y 2}}}}
```

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

vs.

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

Subtypes and Update

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

vs.

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

$$\{x_1, \ldots x_n\} \supseteq \{x'_1, \ldots x'_m\}$$
$$x_i = x'_j \Rightarrow \tau_i \leq \tau'_j$$

$$\{[x_1 : \tau_1] \ldots [x_n : \tau_n]\} \leq \{[x'_1 : \tau'_1] \ldots [x'_m : \tau'_m]\}$$

Seems ok for both functional and imperative update...

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {get
   {get
    {f {record {p {record {x 5}
                          {y 6}}}}}
    p}
   y}}
```

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {get
   {get
    {f {record {p {record {x 5}
                          {y 6}}}}}
    p}
   y}}
```

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

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {get
   {get
    {f {record {p {record {x 5}
                          {y 6}}}}}
     p}
    y}}
```

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

vs.

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

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {let {[r : ?
             {record {p {record {x 5}
                                {y 6}}}}]}
    {begin
      {f r}
      {get {get r p} y}}}}
```

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {let {[r : ?
             {record {p {record {x 5}
                                {y 6}}}}]}
    {begin
      {f r}
      {get {get r p} y}}}}
```

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

Subtypes and Update

```
{let {[f : ?
          {lambda {[r : {[p : {[x : num]}]}]}
            {set r p {record {x 10}}}}]}
  {let {[r : ?
             {record {p {record {x 5}
                                {y 6}}}}]}
    {begin
      {f r}
      {get {get r p} y}}}}
```

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

vs.

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

Subtypes and Update

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

vs.

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

$$\{\mathbf{x}_1, \ \ldots \ \mathbf{x}_n\} \supseteq \{\mathbf{x}'_1, \ \ldots \ \mathbf{x}'_m\}$$

$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i \leq \tau'_j$$

$$\{[\mathbf{x}_1 : \tau_1] \ \ldots \ [\mathbf{x}_n : \tau_n]\} \leq \{[\mathbf{x}'_1 : \tau'_1] \ \ldots \ [\mathbf{x}'_m : \tau'_m]\}$$

Wrong for imperative update!

Subtypes and Update

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

vs.

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

$$\{\mathbf{x}_1, \ldots \mathbf{x}_n\} \supseteq \{\mathbf{x}'_1, \ldots \mathbf{x}'_m\}$$

$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i = \tau'_j$$

$$\{[\mathbf{x}_1 : \tau_1] \ldots [\mathbf{x}_n : \tau_n]\} \leq \{[\mathbf{x}'_1 : \tau'_1] \ldots [\mathbf{x}'_m : \tau'_m]\}$$

Invariance

With imperative update:

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

Field types must be ***invariant*** with record types