

# Part I

# Typed Records

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

NEW

NEW

NEW

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

NEW

## Records

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

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

## Records

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

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

## Records

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

The subexpression type

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

means that `get` will succeed for `y`

## Records

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

The subexpression type

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

means that `get` will succeed for `p` then `x`

## Records

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

Same type and 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 ( $\{[x : \text{num}]\} \rightarrow \text{num}$ )

## Records

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

Has type ( $\{[x : \text{num}]\} \rightarrow \{[x : \text{num}]\}$ )

## Records

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

# Typechecking

$$\Gamma \vdash \mathbf{e}_1 : \tau_1 \quad \dots \quad \Gamma \vdash \mathbf{e}_n : \tau_n$$
$$\tau = \{[\mathbf{x}_1 : \tau_1] \dots [\mathbf{x}_n : \tau_n]\}$$

---

$$\Gamma \vdash \{\mathbf{record} \{[\mathbf{x}_1 : \mathbf{e}_1] \dots [\mathbf{x}_n : \mathbf{e}_n]\}\} : \tau$$

$$\Gamma \vdash \mathbf{e} : \{[\mathbf{x}_1 : \tau_1] \dots [\mathbf{x}_n : \tau_n]\}$$
$$\mathbf{x}_i \in \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$$

---

$$\Gamma \vdash \{\mathbf{get} \mathbf{e} \mathbf{x}_i\} : \tau_i$$

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

---

$$\Gamma \vdash \{\mathbf{set} \mathbf{e}_1 \mathbf{x}_i \mathbf{e}_2\} : \{[\mathbf{x}_1 : \tau_1] \dots [\mathbf{x}_n : \tau_n]\}$$

## Part 2

## Records and Fields

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

## Records and Fields

`({[x : num]} → num)`

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

## Records and Fields

`({[x : num]} → num)`

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

`{[x : num]}`



## Records and Fields

$(\{[x : \text{num}]\} \rightarrow \text{num})$

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

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

Has type  $\text{num}$

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

## 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

`({[x : num] [y : num]} → num)`

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

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

## Records and Fields

$(\{[x : \text{num}] [y : \text{num}]\} \rightarrow \text{num})$

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

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

**no type** — field order doesn't match

$$\frac{\Gamma \vdash e_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \{e_1 e_2\} : \tau_3}$$

## Records and Fields

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

## Records and Fields

`({ [x : num] } → num)`

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

## Records and Fields

`({[x : num]} → num)`

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

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

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



## Records and Fields

$(\{[x : \text{num}]\} \rightarrow \text{num})$

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

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

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

**no type** — extra fields in argument

$$\frac{\Gamma \vdash e_1 : (\tau_2 \rightarrow \tau_3) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \{e_1 e_2\} : \tau_3}$$

## Subtypes

If  $\tau$  is a **subtype** of  $\tau'$ ,

$$\tau \leq \tau'$$

then an expression of type  $\tau$  can be used in place of an expression of type  $\tau'$

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

## Subtypes

If  $\tau$  is a **subtype** of  $\tau'$ ,

$$\tau \leq \tau'$$

then an expression of type  $\tau$  can be used in place of an expression of type  $\tau'$

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

## Subtypes

If  $\tau$  is a **subtype** of  $\tau'$ ,

$$\tau \leq \tau'$$

then an expression of type  $\tau$  can be used in place of an expression of type  $\tau'$

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

## Subtypes

If  $\tau$  is a **subtype** of  $\tau'$ ,

$$\tau \leq \tau'$$

then an expression of type  $\tau$  can be used in place of an expression of type  $\tau'$

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

# Subtypes

If  $\tau$  is a **subtype** of  $\tau'$ ,

$$\tau \leq \tau'$$

then an expression of type  $\tau$  can be used in place of an expression of type  $\tau'$

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

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

## Subtype Rules

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

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

---

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

$$\text{num} \leq \text{num} \quad \text{bool} \leq \text{bool}$$

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

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

---

$$\Gamma \vdash \{\mathbf{e}_1 \ \mathbf{e}_2\} : \tau_3$$

## Records and Fields

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



## Records and Fields

`({[x : num]} → num)`

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

## Records and Fields

`({[x : num]} → num)`

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

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

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

## Records and Fields

`({[x : num]} → num)`

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

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

`{[x : num] [y : num]} ≤ {[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

`({[y : num] [x : num]} → num)`

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

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

## Records and Fields

$(\{[y : \text{num}] [x : \text{num}]\} \rightarrow \text{num})$

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

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

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

## Records and Fields

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



## Records and Fields

`({[x : num] [y : num]} → num)`

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

## Records and Fields

`({[x : num] [y : num]} → num)`

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

`{[y : num]}`

## Records and Fields

`({[x : num] [y : num]} → num)`

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

`{[y : num]}`

`{[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

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

vs.

$$\{ [p : \{ [x : \text{num}] \}] \}$$
$$\{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \supseteq \{ \mathbf{x}'_1, \dots, \mathbf{x}'_m \}$$
$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i = \tau'_j$$

---

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



## Subtypes in Fields

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

vs.

$$\{ [p : \{ [x : \text{num}] \}] \}$$
$$\{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \supseteq \{ \mathbf{x}'_1, \dots, \mathbf{x}'_m \}$$
$$\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i \leq \tau'_j$$

---

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

## Field Update and Subtypes

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

## Field Update and Subtypes

Original rule:

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

Revised rule:

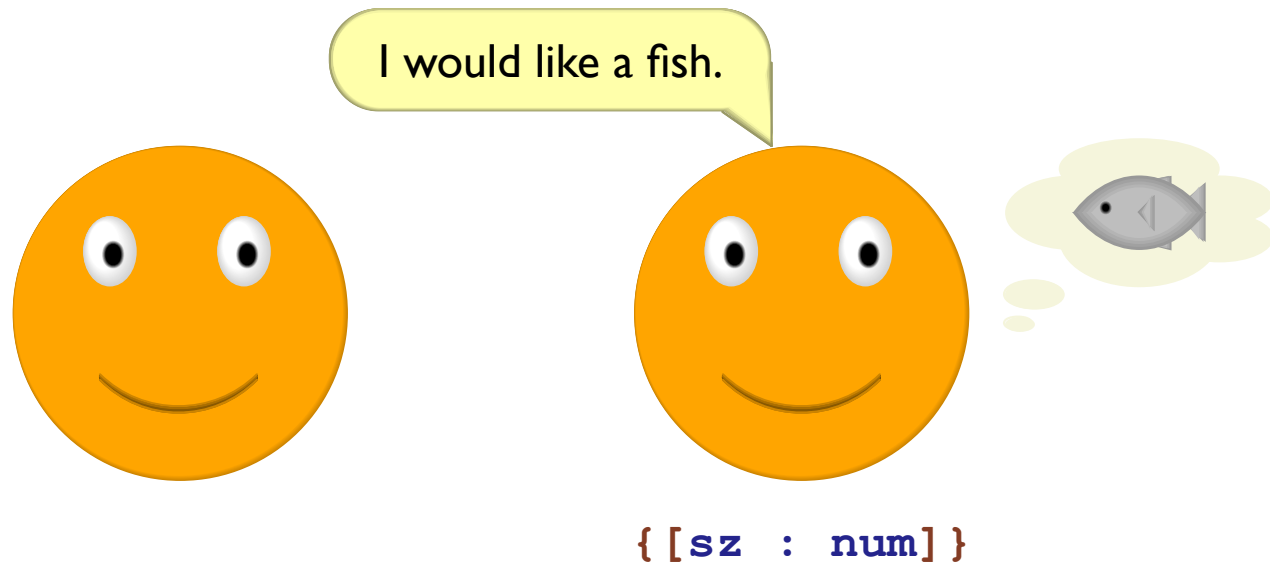
$$\frac{\Gamma \vdash \mathbf{e}_1 : \{[\mathbf{x}_1 : \tau_1] \dots [\mathbf{x}_n : \tau_n]\} \quad \Gamma \vdash \mathbf{e}_2 : \tau' \quad \mathbf{x}_i \in \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \quad \tau' \leq \tau_i}{\Gamma \vdash \{\mathbf{set} \ \mathbf{e}_1 \ \mathbf{x}_i \ \mathbf{e}_2\} : \{[\mathbf{x}_1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

## Part 4

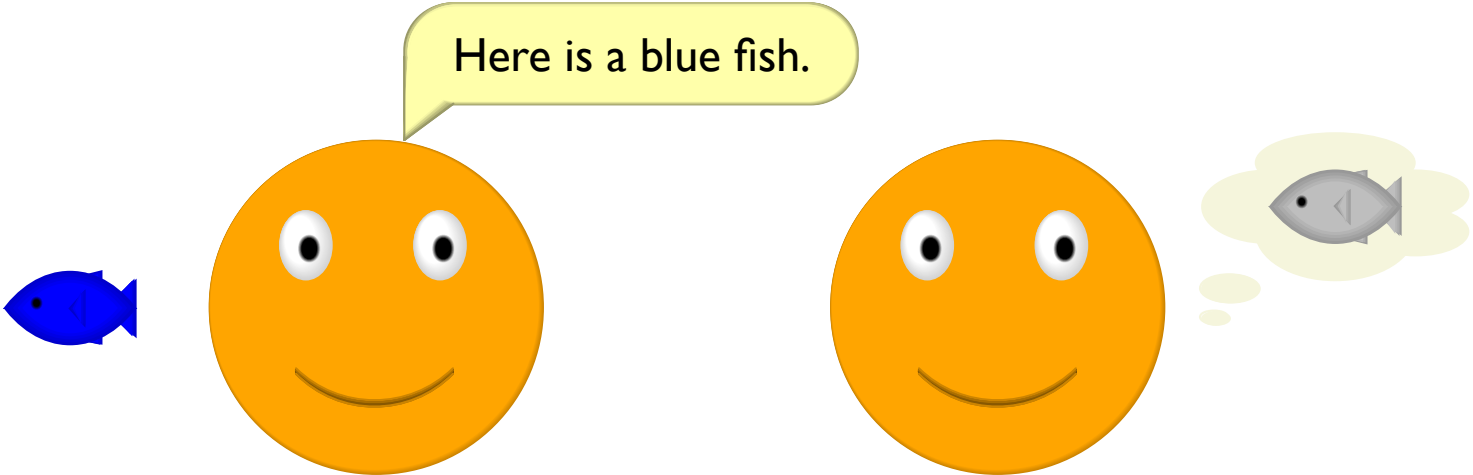
## Subtypes and Functions



## Subtypes and Functions



# Subtypes and Functions



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

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

## Subtypes and Functions



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

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

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

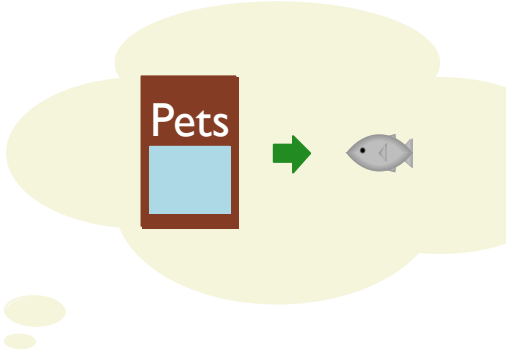


## Subtypes and Functions



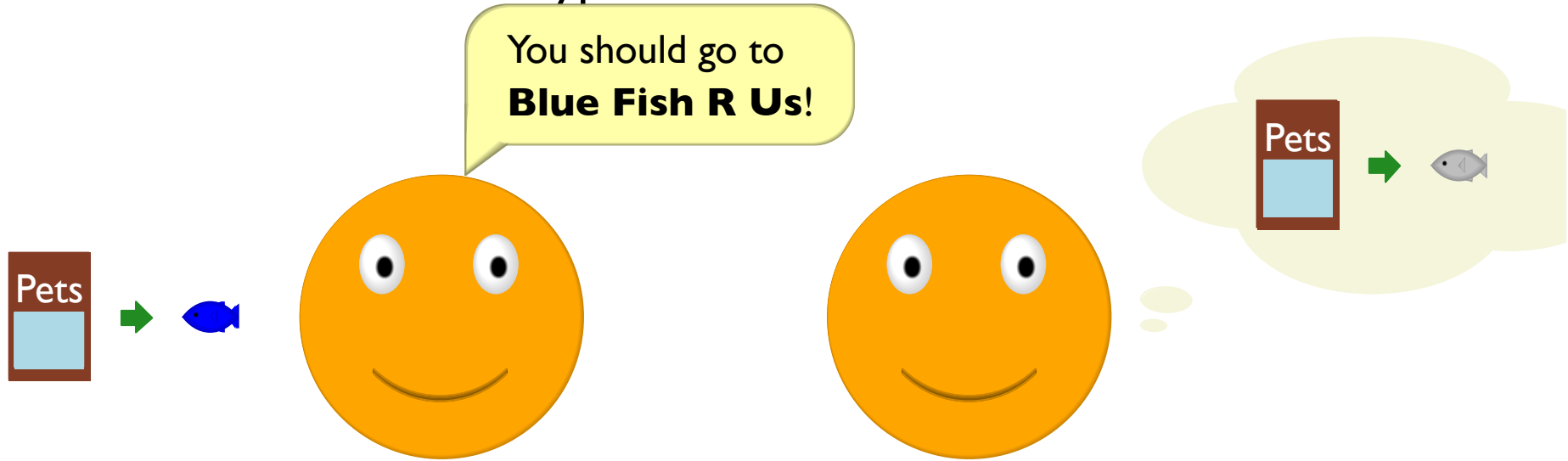
# Subtypes and Functions

Can you recommend a fish store?



```
(num → {[sz : num]})
```

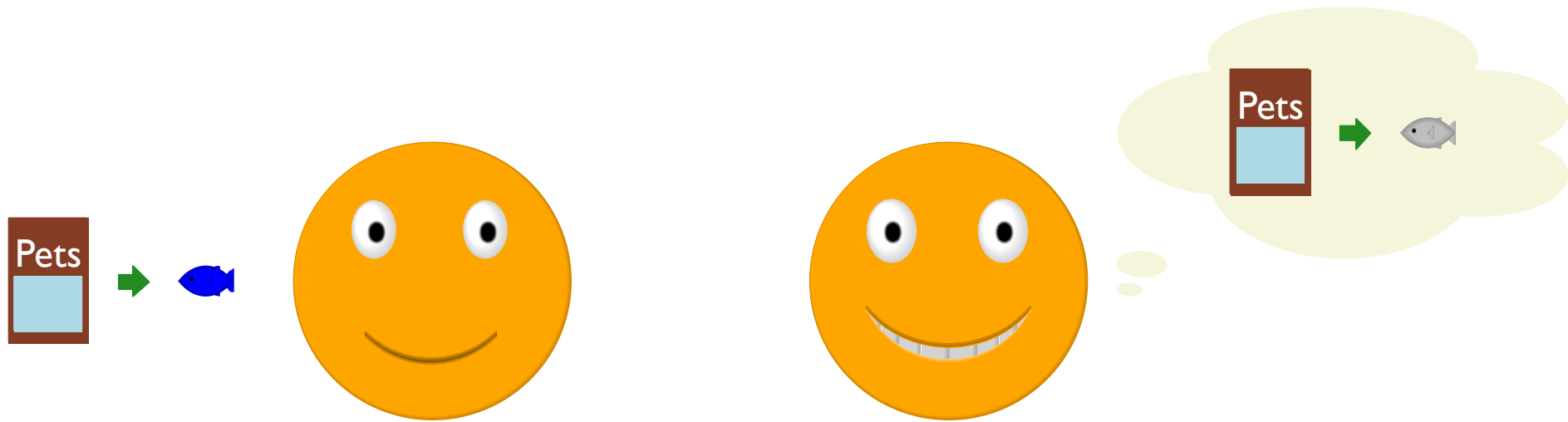
## Subtypes and Functions



`(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]})`

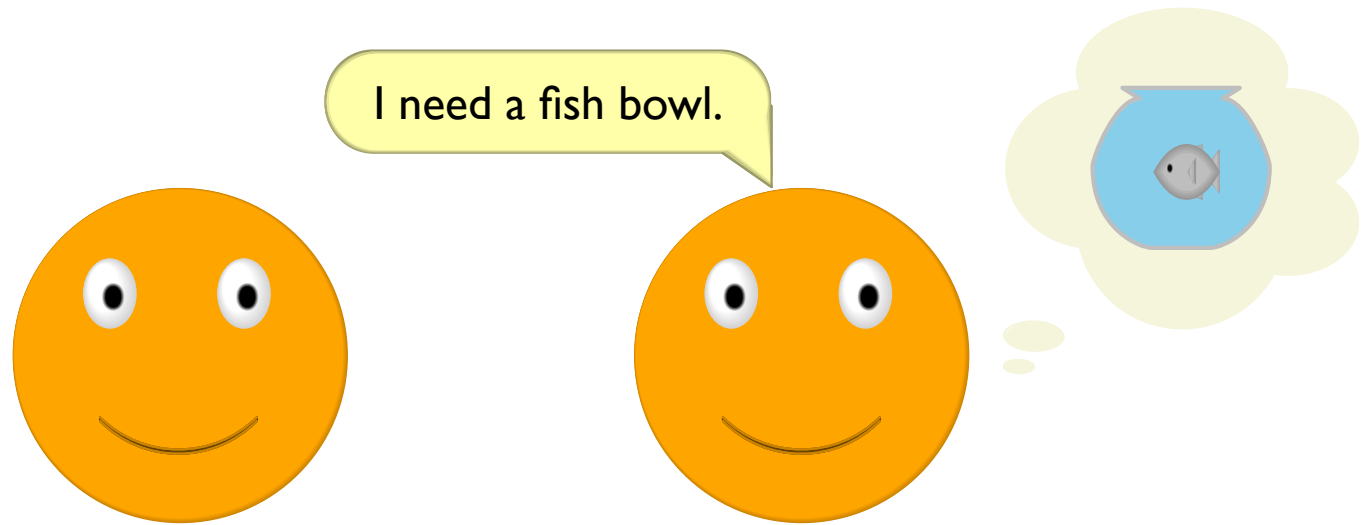
$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \rightarrow \tau_2) \leq (\tau_1 \rightarrow \tau'_2)}$$

## Part 5

## Subtype and Function Arguments

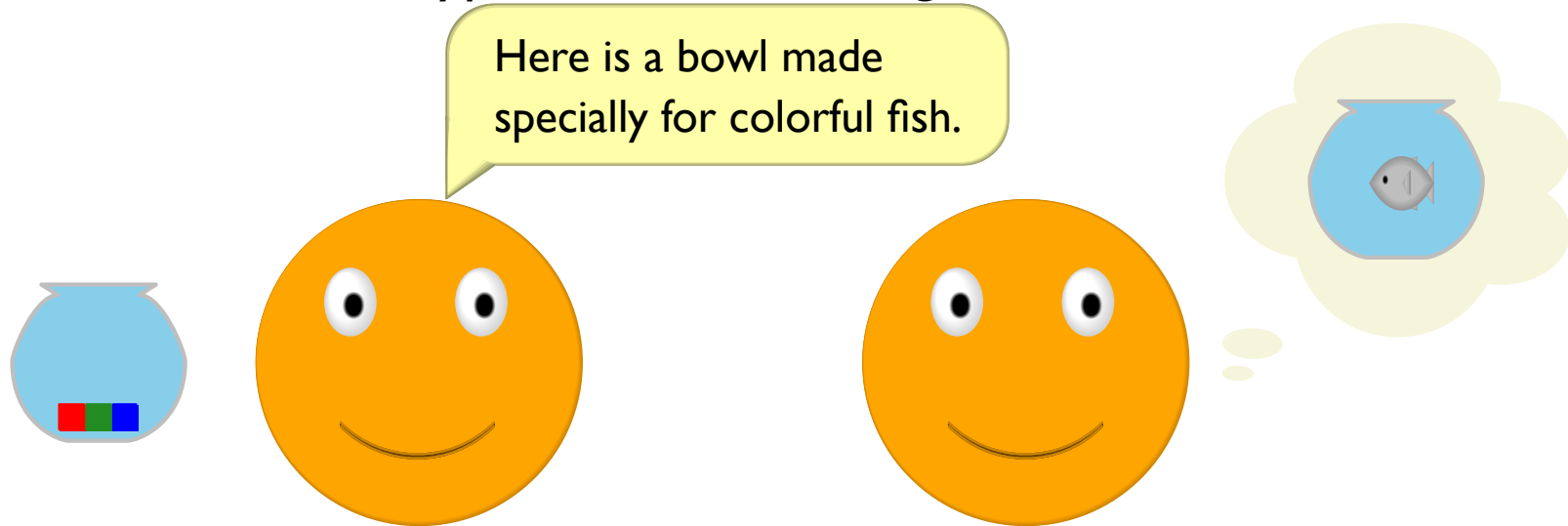


## Subtype and Function Arguments



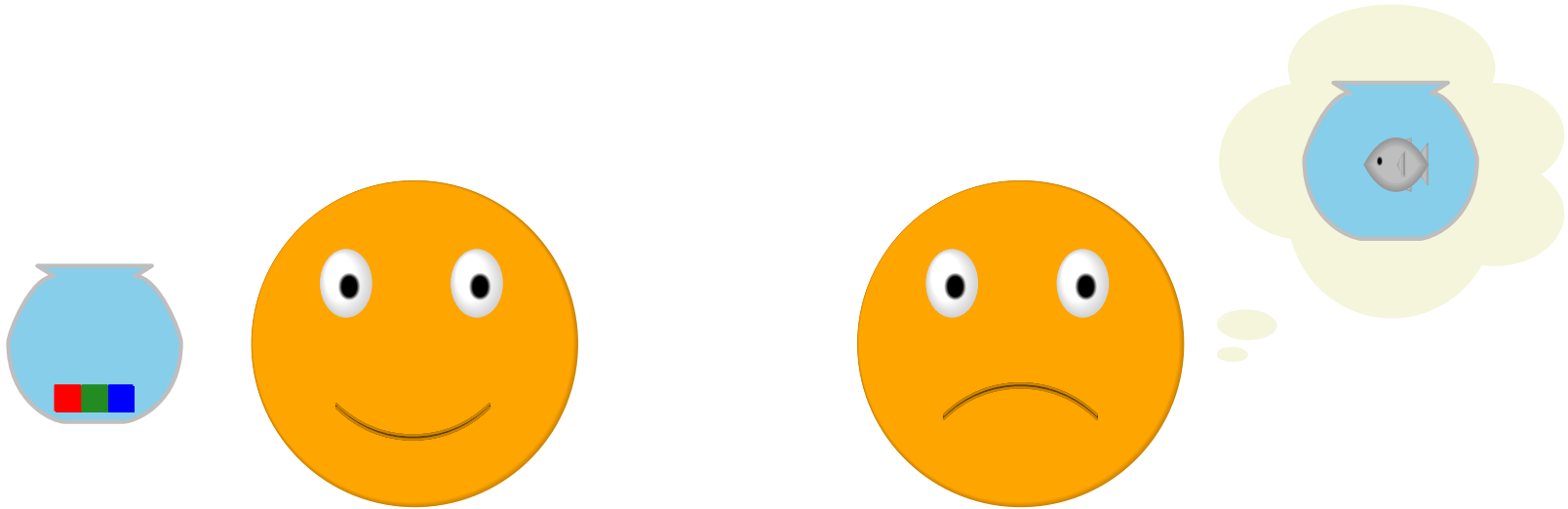
`({ [sz : num] } → num)`

## Subtype and Function Arguments



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

## Subtype and Function Arguments



`({ [sz : num] } → num)`

`({ [sz : num] [col : num] } → num)`

`({ [sz : num] [col : num] } → num)  $\not\leq$  ({ [sz : 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}}}}}
```

## 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)`

$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \rightarrow \tau_2) \leq (\tau_1 \rightarrow \tau'_2)}$$

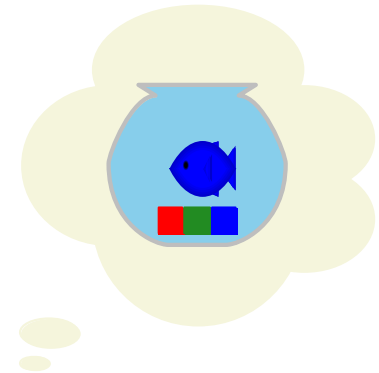
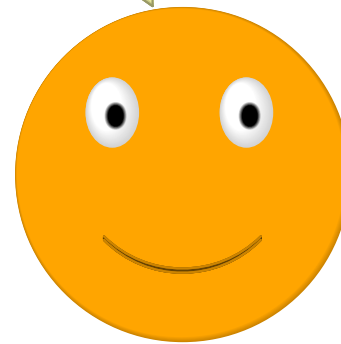
*Correctly rejected!*

## Subtype and Function Arguments



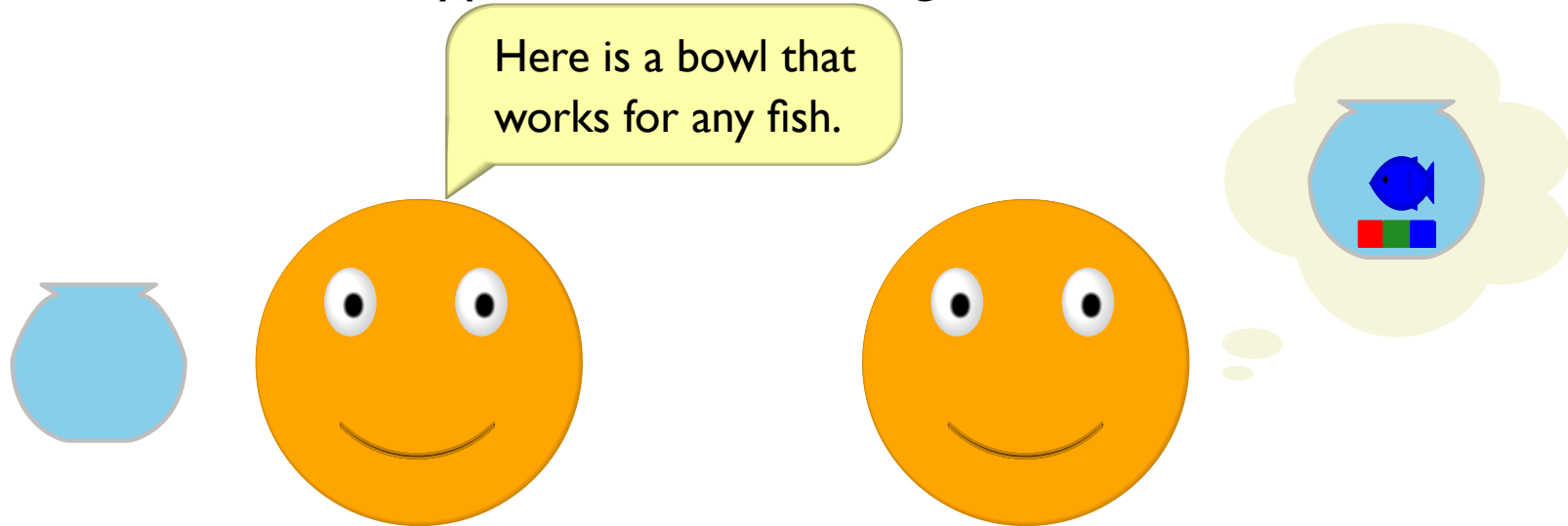
## Subtype and Function Arguments

I need a fish bowl for my blue fish.



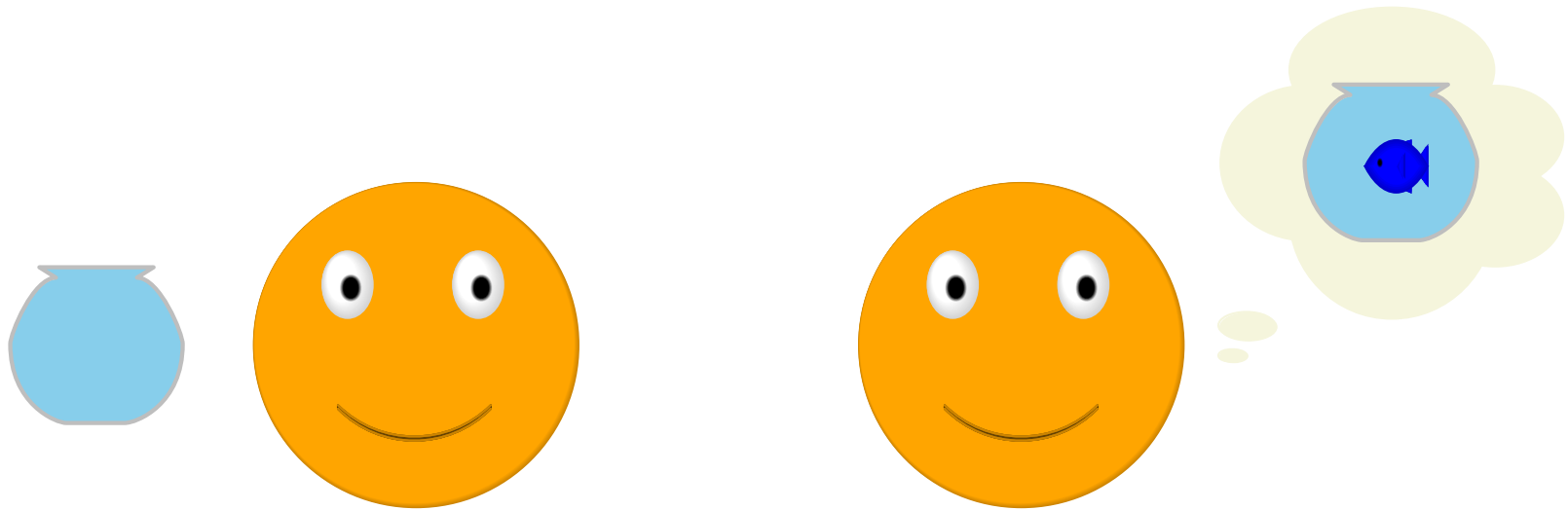
```
({[sz : num] [col : num]} → num)
```

## 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

$(\{ [x : \text{num}] \} \rightarrow \text{num})$

vs.

$(\{ [x : \text{num}] [y : \text{num}] \} \rightarrow \text{num})$

$$\frac{\tau_2 \leq \tau'_2}{(\tau_1 \rightarrow \tau_2) \leq (\tau_1 \rightarrow \tau'_2)}$$

## Subtypes to Functions

$(\{ [x : \text{num}] \} \rightarrow \text{num})$

vs.

$(\{ [x : \text{num}] [y : \text{num}] \} \rightarrow \text{num})$

$$\frac{\tau'_1 \leq \tau_1 \quad \tau_2 \leq \tau'_2}{(\tau_1 \rightarrow \tau_2) \leq (\tau'_1 \rightarrow \tau'_2)}$$

## Part 6

## Covariance and Contravariance

$$\frac{\tau'_1 \preceq \tau_1 \quad \tau_2 \preceq \tau'_2}{(\tau_1 \rightarrow \tau_2) \preceq (\tau'_1 \rightarrow \tau'_2)}$$

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

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

## Covariance and Contravariance

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

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

---

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

Field types are **covariant** with record types

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

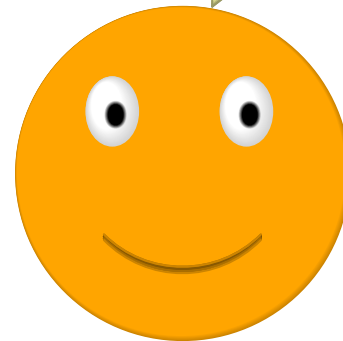
## Part 7

## Subtypes and Functions

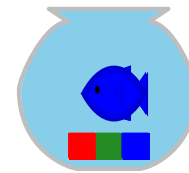




## Subtypes and Functions



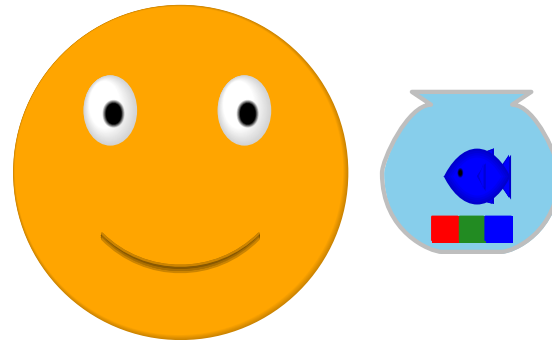
Can you take care of my fish in a bowl while I'm on vacation?.



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

## Subtypes and Functions

Give me a bowl with a fish, and I'll return a bowl with a fish when you get back.

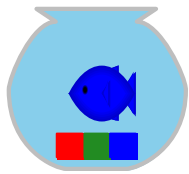


```
{[fish : {[sz : num]  
  [col : num]}}]}
```

```
( {[fish : {[sz : num]}} )  
-> {[fish : {[sz : num]}} )
```

## Subtypes and Functions

Give me a bowl with a fish, and I'll return a bowl with a fish when you get back.

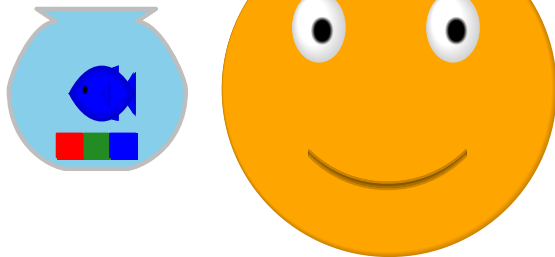


```
{[fish : {[sz : num]  
          [col : num]}}]}
```

```
( {[fish : {[sz : num]}} ]  
  -> {[fish : {[sz : num]}} )
```

## Subtypes and Functions

Give me a bowl with a fish, and I'll return a bowl with a fish when you get back.

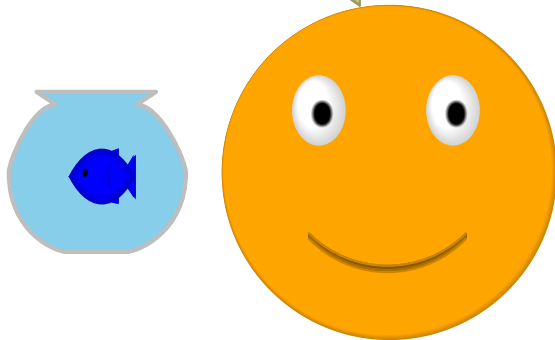


```
{[fish : {[sz : num]
          [col : num]}}]}

({[fish : {[sz : num]}}]}
 -> {[fish : {[sz : num]}}])
```

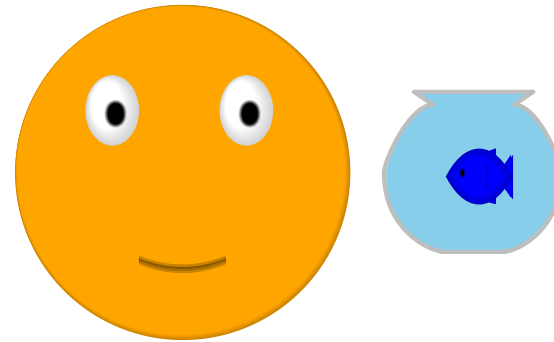
## Subtypes and Functions

Let's move the fish to this other bowl.



```
{[fish : {[sz : num]
          [col : num]}}]
([[fish : {[sz : num]}])
-> {[fish : {[sz : num]}])
```

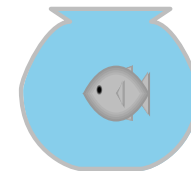
## Subtypes and Functions



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

```
( {[fish : {[sz : num]}] }  
  -> {[fish : {[sz : num]}] } )
```

## Subtypes and Functions



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

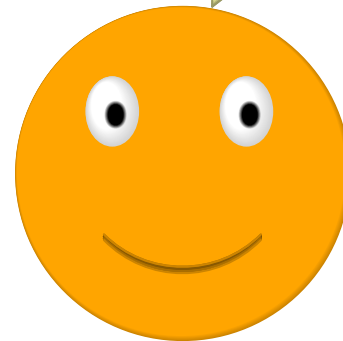
```
( {[fish : {[sz : num]}} )  
-> {[fish : {[sz : num]}} )
```

## Subtypes and Functions

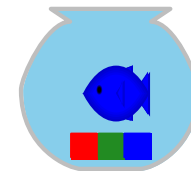




## Subtypes and Functions



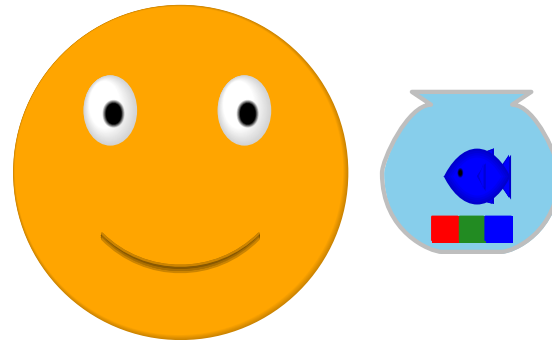
Can you take care of the fish in my bowl while I'm on vacation?.



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

## Subtypes and Functions

Give me a bowl with a fish, and I'll take care of it.

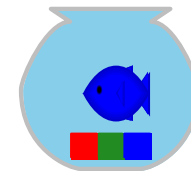


```
{[fish : {[sz : num]
          [col : num]}]}
```

```
( {[fish : {[sz : num]}] }
  -> bool)
```

## Subtypes and Functions

Give me a bowl with a fish, and I'll take care of it.

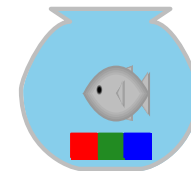


```
{[fish : {[sz : num]
          [col : num]}}]}

({[fish : {[sz : num]}}]}
-> bool)
```

## Subtypes and Functions

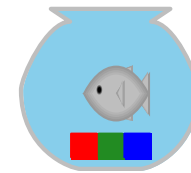
I like this gray fish better!



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

```
( {[fish : {[sz : num]}} ]  
-> bool)
```

## Subtypes and Functions



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

```
( {[fish : {[sz : num]}] }  
-> bool)
```

## 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

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

vs.

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

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

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

---

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

*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

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

vs.

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

$\{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \supseteq \{ \mathbf{x}'_1, \dots, \mathbf{x}'_m \}$   
 $\mathbf{x}_i = \mathbf{x}'_j \Rightarrow \tau_i \leq \tau'_j$

---

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

*Wrong for imperative update!*



## Subtypes and Update

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

vs.

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

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

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

---

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

# Invariance

With imperative update:

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

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

---

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

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