Automating reasoning in cubical type theory

EPN WG 6 meeting
Vienna, 25 April 2023

Maximilian Doré
maximilian.dore@cs.ox.ac.uk

Motivation

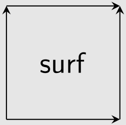
- Cubical Agda introduces new kind of proof obligation: given a boundary in a Kan cubical set, construct a cell with that boundary.
- Explore interval substitutions and Kan compositions as principles of logic
  $\rightarrow$ Automate higher equational reasoning

Higher inductive types and interval substitutions

In Cubical Agda, higher inductive types generate cubical sets.

data Sphere **where**

- base : Sphere
- surf : Path (Path Sphere base base) refl refl



Cells can be contorted with interval substitutions: Abstracting over interval variables drags out a cube, application of formulas to cells prescribe the boundary of the produced cube.

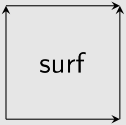
$$\begin{aligned}
&\lambda\ i\ j\ k \to \mathsf{surf}\ (i \vee k)\ ((i \wedge j) \vee k))\\
&\quad : \mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path\ Sphere}\\
&\qquad (\mathsf{surf}\ i\ (i \wedge j))\ \mathsf{base})\\
&\qquad (\lambda k \to \mathsf{surf}\ (i \vee k)\ k)\ (\lambda k \to \mathsf{surf}\ (i \vee k)\ (i \vee k)))\\
&\qquad (\lambda j\ k \to \mathsf{surf}\ k\ k)\ (\lambda j\ k \to \mathsf{base})
\end{aligned}$$

Interval substitutions as poset maps

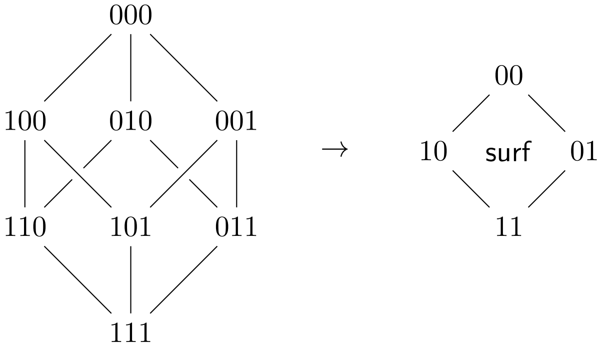
data Sphere **where**

- base : Sphere
- surf : Path (Path Sphere base base) refl refl

$\lambda\ i\ j\ k \to \mathsf{surf}\ (i \vee k)\ ((i \wedge j) \vee k))$



i	j	k	$i \vee k$	$(i \wedge j) \vee k$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

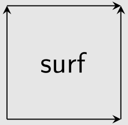


Interval substitutions as poset maps

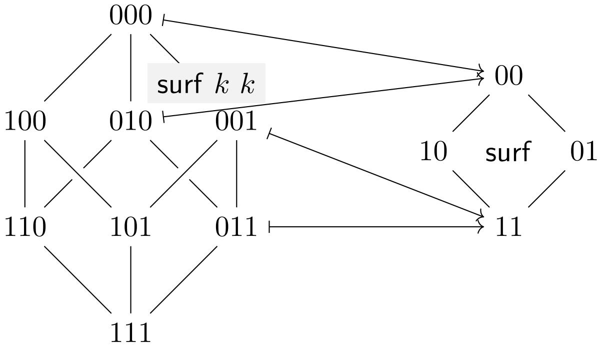
data Sphere **where**

- base : Sphere
- surf : Path (Path Sphere base base) refl refl

$\lambda\ i\ j\ k \to$ surf $(i \vee k)\ ((i \wedge j) \vee k))$



i	j	k	$i \vee k$	$(i \wedge j) \vee k$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

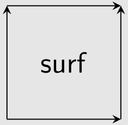


Interval substitutions as poset maps

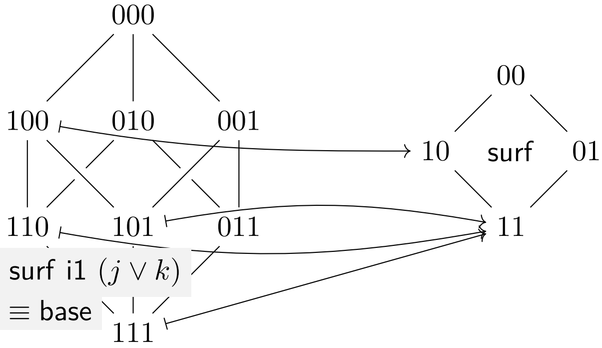
data Sphere **where**

- base : Sphere
- surf : Path (Path Sphere base base) refl refl

$\lambda\ i\ j\ k \to$ surf $(i \vee k)\ ((i \wedge j) \vee k))$



i	j	k	$i \vee k$	$(i \wedge j) \vee k$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

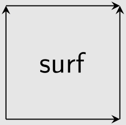


Interval substitutions as poset maps

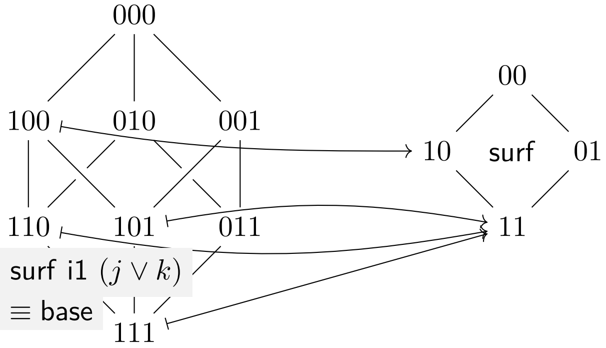
data Sphere **where**

- base : Sphere
- surf : Path (Path Sphere base base) refl refl

$\lambda\ i\ j\ k \to \mathsf{surf}\ (i \vee k)\ ((i \wedge j) \vee k))$



i	j	k	$i \vee k$	$(i \wedge j) \vee k$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Higher-dimensional faces determine the poset map the most.

Cubical sets

$\square_{\wedge\vee}$ is the full subcategory of the category of posets and monotone maps with objects $\mathbf{I}^n$ for $n \geq 0$, where $\mathbf{I} = \{0 < 1\}$.

A cubical set is an object of $\mathbf{Set}^{\square_{\wedge\vee}{}^{op}}$.

All morphisms in $\square_{\wedge\vee}$ are of the form $\mathbf{I}^m \to \mathbf{I}^n$, e.g.:

$$s^i : \mathbf{I}^n \to \mathbf{I}^{n-1}, (e_1 \dots e_n) \mapsto (e_1 \dots e_{i-1} e_{i+1} \dots e_n) \text{ for } 1 \leq i \leq n$$

$$d^{(i,e)} : \mathbf{I}^{n-1} \to \mathbf{I}^n, (e_1 \dots e_{n-1}) \mapsto (e_1 \dots e_{i-1} e e_{i+1} \dots e_{n-1}) \text{ for } 1 \leq i \leq n, e \in \{0, 1\}$$

Given an n-cell p of a cubical set X and a poset map $\sigma : \mathbf{I}^m \to \mathbf{I}^n$, call $p\langle\sigma\rangle := X(\sigma)(p)$ an m-contortion of p.

Its boundary is $\partial(p) := [p\langle d^{(1,\mathbf{0})}\rangle, p\langle d^{(1,\mathbf{1})}\rangle, \dots p\langle d^{(n,\mathbf{0})}\rangle, p\langle d^{(n,\mathbf{1})}\rangle]$

Cubical sets

$\square_{\wedge\vee}$ is the full subcategory of the category of posets and monotone maps with objects $\mathbf{I}^n$ for $n \geq 0$, where $\mathbf{I} = \{0 < 1\}$.

A <u>cubical set</u> is an object of $\mathbf{Set}^{\square_{\wedge\vee}{}^{op}}$.

All morphisms in $\square_{\wedge\vee}$ are of the form $\mathbf{I}^m \to \mathbf{I}^n$, e.g.:

$$s^i : \mathbf{I}^n \to \mathbf{I}^{n-1}, (e_1 \dots e_n) \mapsto (e_1 \dots e_{i-1} e_{i+1} \dots e_n) \text{ for } 1 \leq i \leq n$$

$$d^{(i,e)} : \mathbf{I}^{n-1} \to \mathbf{I}^n, (e_1 \dots e_{n-1}) \mapsto (e_1 \dots e_{i-1} e e_{i+1} \dots e_{n-1}) \quad \text{for } 1 \leq i \leq n, e \in \{0,1\}$$

Given an n-cell p of a cubical set X and a poset map $\sigma : \mathbf{I}^m \to \mathbf{I}^n$, call <u>$p\langle\sigma\rangle := X(\sigma)(p)$</u> an m-contortion of p.

Its boundary is $\partial(p) := [p\langle d^{(1,\mathbf{0})}\rangle, p\langle d^{(1,\mathbf{1})}\rangle, \dots p\langle d^{(n,\mathbf{0})}\rangle, p\langle d^{(n,\mathbf{1})}\rangle]$

<u>Problem **CubicalCell**</u>: given a boundary T and an n-cell p, find a poset map $\sigma : \mathbf{I}^m \to \mathbf{I}^n$ such that $\partial(p\langle\sigma\rangle) = T$.

Representing the search space

Represent a collection of poset maps as follows:

A potential poset map (ppm) is a map $\Sigma : \mathbf{I}^m \to \mathcal{P}(\mathbf{I}^n)$ such that $\forall x \leq y$:

- $\forall u \in \Sigma(y) : \exists v \in \Sigma(x) : v \leq u$.
- $\forall v \in \Sigma(x) : \exists u \in \Sigma(y) : v \leq u$.

Given $x \in \mathbf{I}^m$, any $y \in \Sigma(x)$ induces at least one $\sigma : \mathbf{I}^m \to \mathbf{I}^n$.

Total ppm $\Sigma(x) \mapsto \mathbf{I}^n$ grows exponentially in m and n.

Representing the search space

Represent a collection of poset maps as follows:

A potential poset map (ppm) is a map $\Sigma : \mathbf{I}^m \to \mathcal{P}(\mathbf{I}^n)$ such that $\forall x \leq y$:

- $\forall u \in \Sigma(y) : \exists v \in \Sigma(x) : v \leq u.$
- $\forall v \in \Sigma(x) : \exists u \in \Sigma(y) : v \leq u.$

Given $x \in \mathbf{I}^m$, any $y \in \Sigma(x)$ induces at least one $\sigma : \mathbf{I}^m \to \mathbf{I}^n$.

Total ppm $\Sigma(x) \mapsto \mathbf{I}^n$ grows exponentially in m and n. 🎉

Searching for contortions

Given an n-cell p and an m-dimensional boundary T.

- start with total ppm $\Sigma(x) = \mathbf{I}^n$ for all x.
- for each $q\langle\tau\rangle = T_{i,e}$ with $\mathsf{dim}(q)$ decreasing:
  - if $q = p$, then $\Sigma(x) = \{\tau(x)\}$ for all $x \in \mathbf{I}^n$.
  - o/w $\Sigma(x) = \{y \mid \exists\sigma' \in \Sigma \text{ with } \sigma'(x) = y \text{ s.t. } p\langle\sigma'\rangle = T_{(i=e)}\}$
- return $\sigma \in \Sigma$ such that $\partial(p\langle\sigma\rangle) = T$.

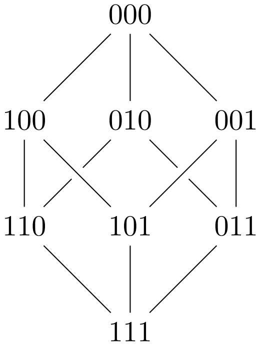
Constructing a poset map

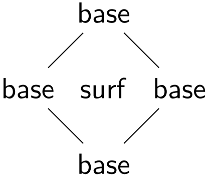
$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})\\ (\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

Constructing a poset map

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})$$
$$(\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

Goal: $[\mathsf{surf}\langle \begin{smallmatrix} 00 \mapsto 00 \\ 01 \mapsto 01 \\ 10 \mapsto 01 \\ 11 \mapsto 11 \end{smallmatrix} \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle]$

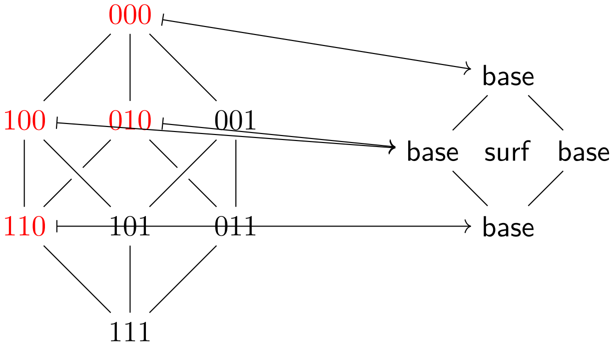




Constructing a poset map

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})\\ (\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

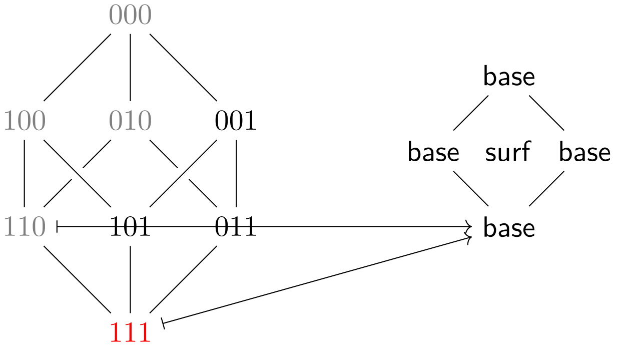
Goal: $[\mathsf{surf}\langle{}^{00\mapsto 00}_{\substack{01\mapsto 01\\10\mapsto 01\\11\mapsto 11}}\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle]$



Constructing a poset map

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})$$
$$(\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

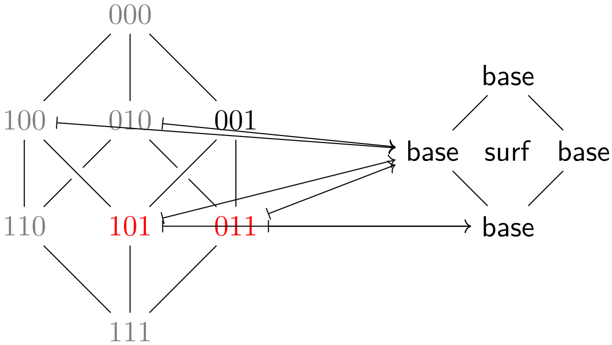
Goal: $[\mathsf{surf}\langle^{00\mapsto 00}_{\substack{01\mapsto 01\\ 10\mapsto 01}}{}_{11\mapsto 11}\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle]$



Constructing a poset map

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})$$
$$(\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

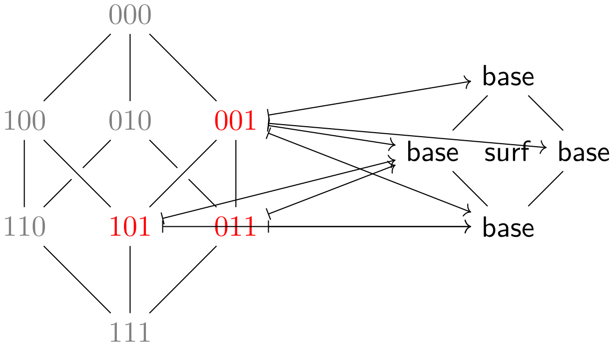
Goal: $[\mathsf{surf}\langle {}^{00 \mapsto 00}_{\substack{01 \mapsto 01 \\ 10 \mapsto 01}}{}_{11 \mapsto 11} \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle, \mathsf{base}\langle s^2 \rangle]$



Constructing a poset map

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{surf}\ (i \wedge j)\ (i \vee j))\ \mathsf{base})$$
$$(\lambda k \to \mathsf{base})(\lambda k \to \mathsf{base}))(\lambda jk \to \mathsf{base})(\lambda jk \to \mathsf{base})$$

Goal: $[\mathsf{surf}\langle {}^{00\mapsto 00}_{01\mapsto 01}{}^{}_{10\mapsto 01}{}_{11\mapsto 11} \rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle, \mathsf{base}\langle s^2\rangle]$



Complexity of contortion search

When checking whether we can contort an n-cell into an m-dimensional boundary, we have to evaluate $\mathcal{O}(2mD^n_{m-1})$ many contortions – bruteforce would require $\mathcal{O}(D^n_m)$.

In many cases we have to check significantly fewer.

Complexity of contortion search

When checking whether we can contort an n-cell into an m-dimensional boundary, we have to evaluate $\mathcal{O}(2mD^n_{m-1})$ many contortions – bruteforce would require $\mathcal{O}(D^n_m)$.

In many cases we have to check significantly fewer.

6-dim analogue of goal requires checking < 16.000 poset maps.

```
λ> solve sphere sphere6Cube
Just (Term "p" (fromList [(000000,00),(000001,01),(000010,01),(000011,01),(000100,01),(000101,01),(
000110,01),(000111,01),(001000,01),(001001,01),(001010,01),(001011,01),(001100,01),(001101,01),(001
110,01),(001111,01),(010000,01),(010001,01),(010010,01),(010011,01),(010100,01),(010101,01),(010110
,01),(010111,01),(011000,01),(011001,01),(011010,01),(011011,01),(011100,01),(011101,01),(011110,01
),(011111,11),(100000,01),(100001,01),(100010,01),(100011,01),(100100,01),(100101,01),(100110,01),(
100111,01),(101000,01),(101001,01),(101010,01),(101011,01),(101100,01),(101101,01),(101110,01),(101
111,01),(110000,01),(110001,01),(110010,01),(110011,01),(110100,01),(110101,01),(110110,01),(110111
,01),(111000,01),(111001,01),(111010,01),(111011,01),(111100,01),(111101,01),(111110,01),(111111,11
)]))
(1.85 secs, 1,284,512,368 bytes)
λ> (putStrLn . agdaShow . fromJust) (solve sphere sphere6Cube)
p (j ∧ k ∧ l ∧ m ∧ n) (i ∨ j ∨ k ∨ l ∨ m ∨ n)
(1.82 secs, 1,283,986,024 bytes)
```

Brute force: $D_6^2 = 7.828.354^2 = 61.283.126.349.316$ poset maps.

Kan cubical sets

Crucial reasoning principle in Cubical Agda: hcomp

An $(n+1)$-dimensional open box is a collection of $2n+1$ cells $[t_{1,\mathbf{0}}, t_{1,\mathbf{1}}, \ldots, t_{n,\mathbf{0}}, t_{n,\mathbf{1}}]u$ such that

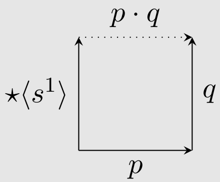
- $t_{i,e}\langle d^{(n,\mathbf{0})}\rangle = u\langle d^{(i,e)}\rangle$ for all $1 \leq i \leq n$, $e \in \{\mathbf{0}, \mathbf{1}\}$
- $t_{i,e}\langle d^{(j,e')}\rangle = t_{j,e'}\langle d^{(i,e)}\rangle$ for $1 \leq i < j \leq n$ and $e, e \in \{\mathbf{0}, \mathbf{1}\}$.

A Kan cubical set has for any open box U a front side Comp U.

data Paths **where**

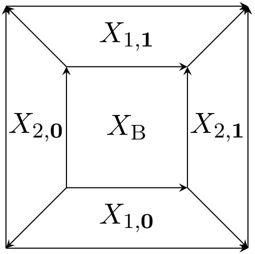
- $\star$: Paths
- $p, q, r : \star = \star$

$p \cdot q :=$ Comp $[\star\langle s^1\rangle, q]p$

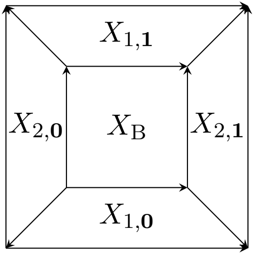


Finding open boxes as a constraint satisfaction problem

Problem **KanCubicalCell**: given a boundary T, find an open cube U with front T.



Finding open boxes as a constraint satisfaction problem



<u>Problem **KanCubicalCell**</u>: given a boundary T, find an open cube U with front T.

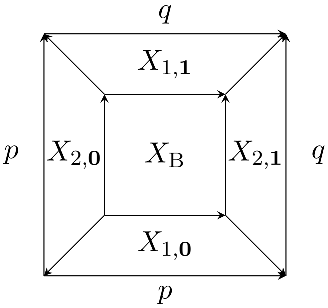
- Variables X_{B} and $X_{i,\mathbf{0}}$, $X_{i,\mathbf{1}}$ for $1 \leq i \leq n$
- Domains
  - $D_{\mathrm{B}} = \{p\langle\Sigma\rangle \mid p \in \Gamma\}$
  - $D_{i,e} = \{p\langle\Sigma\rangle \mid p \in \Gamma, p\langle\Sigma\rangle\langle d^{(n,\mathbf{1})}\rangle = T_{i,e}\}$
- Constraints
  - $X_{i,e}\langle d^{(n,\mathbf{0})}\rangle = X_{\mathrm{B}}\langle d^{(i,e)}\rangle$ for all $1 \leq i \leq n$, $e \in \{\mathbf{0}, \mathbf{1}\}$
  - $X_{i,e}\langle d^{(j,e')}\rangle = X_{j,e'}\langle d^{(i,e)}\rangle$ for $1 \leq i < j \leq n$, $e, e \in \{\mathbf{0}, \mathbf{1}\}$.

Simple Kan composition

data Paths **where**

- $\star$: Paths
- $p, q : \star = \star$

Goal: $[p, q, p, q]$



$$D_{1,0} = \{p\langle\begin{smallmatrix} 00\mapsto 0 \\ 01\mapsto 0,1 \\ 10\mapsto 0 \\ 11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,1} = \{q\langle\begin{smallmatrix} 00\mapsto 0 \\ 01\mapsto 0,1 \\ 10\mapsto 0 \\ 11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,0} = \{p\langle\begin{smallmatrix} 00\mapsto 0 \\ 01\mapsto 0,1 \\ 10\mapsto 0 \\ 11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,1} = \{q\langle\begin{smallmatrix} 00\mapsto 0 \\ 01\mapsto 0,1 \\ 10\mapsto 0 \\ 11\mapsto 1\end{smallmatrix}\rangle\}$$

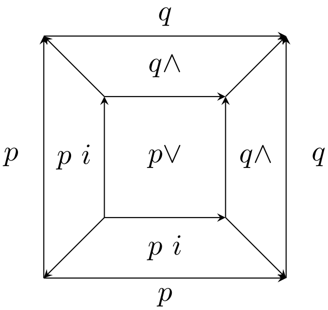
$$D_B = \{\star\langle\begin{smallmatrix} 00\mapsto () \\ 01\mapsto () \\ 10\mapsto () \\ 11\mapsto ()\end{smallmatrix}\rangle, p\langle\begin{smallmatrix} 00\mapsto 0,1 \\ 01\mapsto 0,1 \\ 10\mapsto 0,1 \\ 11\mapsto 0,1\end{smallmatrix}\rangle, q\langle\begin{smallmatrix} 00\mapsto 0,1 \\ 01\mapsto 0,1 \\ 10\mapsto 0,1 \\ 11\mapsto 0,1\end{smallmatrix}\rangle\}$$

Simple Kan composition

data Paths **where**

- $\star$: Paths
- $p, q : \star = \star$

Goal: $[p, q, p, q]$



$$D_{1,0} = \{p\langle\begin{smallmatrix}00\mapsto 0\\01\mapsto 1\\10\mapsto 0\\11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,1} = \{q\langle\begin{smallmatrix}00\mapsto 0\\01\mapsto 0\\10\mapsto 0\\11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,0} = \{p\langle\begin{smallmatrix}00\mapsto 0\\01\mapsto 1\\10\mapsto 0\\11\mapsto 1\end{smallmatrix}\rangle\}$$

$$D_{1,1} = \{q\langle\begin{smallmatrix}00\mapsto 0\\01\mapsto 0\\10\mapsto 0\\11\mapsto 1\end{smallmatrix}\rangle\}$$

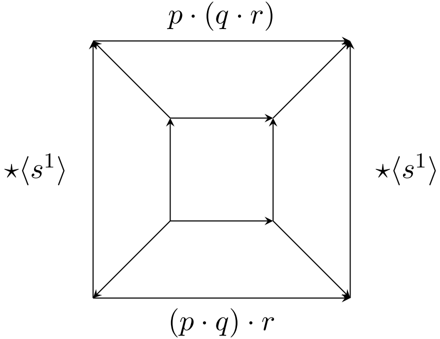
$$D_B = \{p\langle\begin{smallmatrix}00\mapsto 0\\01\mapsto 1\\10\mapsto 1\\11\mapsto 1\end{smallmatrix}\rangle\}$$

Associativity of path composition

data Paths **where**

- $\star$: Paths
- $p, q, r : \star = \star$

Goal: $(p \cdot q) \cdot r = p \cdot (q \cdot r) \quad \rightsquigarrow \quad [(p \cdot q) \cdot r, p \cdot (q \cdot r), \star\langle s^1 \rangle, \star\langle s^1 \rangle]$

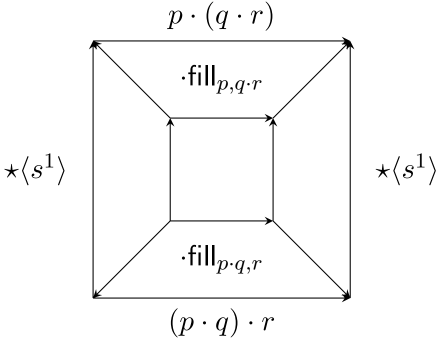


Associativity of path composition

data Paths **where**

- $\star : \mathsf{Paths}$
- $p, q, r : \star = \star$

Goal: $(p \cdot q) \cdot r = p \cdot (q \cdot r) \quad \rightsquigarrow \quad [(p \cdot q) \cdot r, p \cdot (q \cdot r), \star\langle s^1 \rangle, \star\langle s^1 \rangle]$



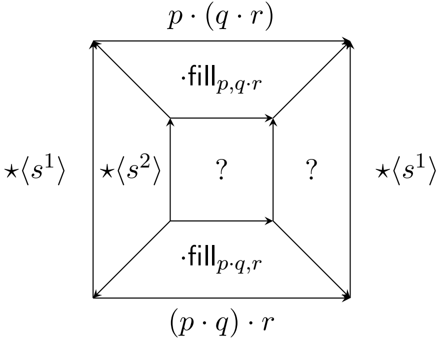
Unfold Kan compositions of goal boundary

Associativity of path composition

data Paths **where**

- $\star$: Paths
- $p, q, r : \star = \star$

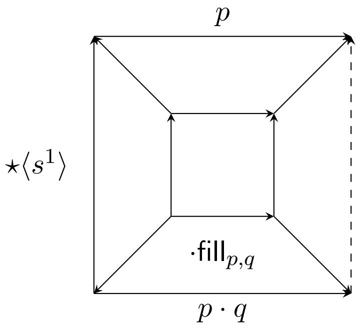
Goal: $(p \cdot q) \cdot r = p \cdot (q \cdot r) \quad \rightsquigarrow \quad [(p \cdot q) \cdot r, p \cdot (q \cdot r), \star\langle s^1 \rangle, \star\langle s^1 \rangle]$



Fill sides with contortions if possible

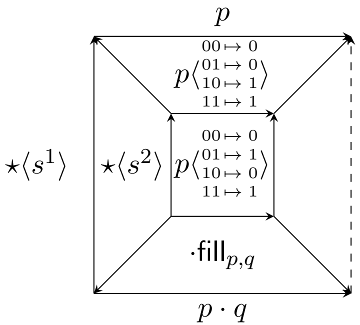
Filling the open sides

Filler for the back:



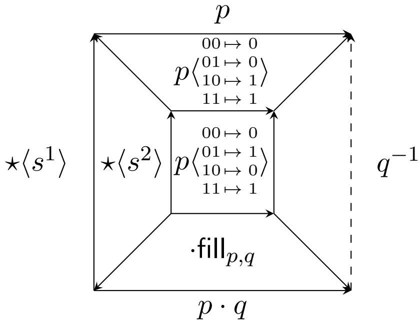
Filling the open sides

Filler for the back:

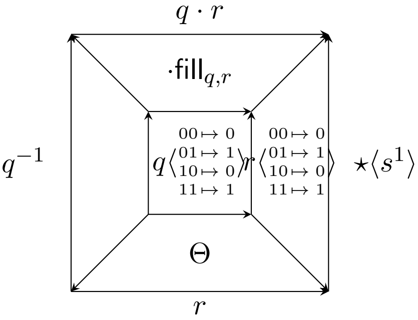


Filling the open sides

Filler for the back:



Filler for the right side:



Open faces can be filled with Kan fillers

Kan composition algorithm

Given goal T, construct a nested Kan composition as follows:

- Solve CSP for open box with front boundary T
  - Unfold Kan compositions if possible
  - Fill as many sides with contortions as possible
  - If not all sides can be filled, use Kan fillers for open faces
- Call composition solver on open sides of the cube

Complete calculus. CSPs stay small, not much memory needed.

Proving Eckmann-Hilton

data EckmannHilton **where**

- $\star$: EckmannHilton
- p, q : Path (Path EckmannHilton $\star$ $\star$) refl refl

Goal: $p \cdot q = q \cdot p \quad \rightsquigarrow$
$[\mathsf{Comp}\,[\star\langle s^1\rangle, q, \star\langle s^1\rangle, \star\langle s^1\rangle]p, \mathsf{Comp}\,[\star\langle s^1\rangle, p, \star\langle s^1\rangle, \star\langle s^1\rangle]q,$
$\star\langle s^2\rangle, \star\langle s^2\rangle, \star\langle s^2\rangle, \star\langle s^2\rangle]$

Proving Eckmann-Hilton

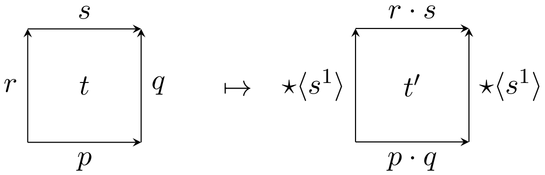
data EckmannHilton **where**

- $\star$: EckmannHilton
- p, q : Path (Path EckmannHilton $\star$ $\star$) refl refl

Goal: $p \cdot q = q \cdot p \quad \rightsquigarrow$
$[\mathsf{Comp}\ [\star\langle s^1\rangle, q, \star\langle s^1\rangle, \star\langle s^1\rangle]p, \mathsf{Comp}\ [\star\langle s^1\rangle, p, \star\langle s^1\rangle, \star\langle s^1\rangle]q,$
$\star\langle s^2\rangle, \star\langle s^2\rangle, \star\langle s^2\rangle, \star\langle s^2\rangle]$

Prove this in two steps:

- Fill the cube $[p, p, q, q, \star\langle s^2\rangle, \star\langle s^2\rangle]$
- Show $t : [p, s, r, q]$ gives rise to $t' : [p \cdot q, r \cdot s, \star\langle s^1\rangle, \star\langle s^1\rangle]$



Conclusions

- The problem of finding Kan cubical cells generalises word problems for various algebraic structures, satisfiability, ...
  $\rightarrow$ derivation *spaces* instead of trees
- Desired takeaways:
  - *Practical*: develop a tactic for automatic proof search. Can we derive new proofs?
  - *Foundational*: even in weak model, can we discard most coherences automatically? Have to leave the garden eden of decidability...

Conclusions

- The problem of finding Kan cubical cells generalises word problems for various algebraic structures, satisfiability, ...
  $\rightarrow$ derivation *spaces* instead of trees
- Desired takeaways:
  - *Practical*: develop a tactic for automatic proof search. Can we derive new proofs?
  - *Foundational*: even in weak model, can we discard most coherences automatically? Have to leave the garden eden of decidability...

Thank you for your attention!

Demos

```
λ i j → hcomp (λ k → λ {
    (i = i0) → _
  ; (i = i1) → _
  ; (j = i0) → a
  ; (j = i1) → λ k l → hcomp (λ m → λ {
        (k = i0) → λ m n → hcomp (λ o → λ
            (m = i0) → q i
          ; (m = i1) → r (i ∧ l)
          ; (n = i0) → q i
          ; (n = i1) → r (i ∧ l)
          }) (q (i ∨ l))
      ; (k = i1) → _
      ; (l = i0) → _
      ; (l = i1) → r k
      }) (q k)
  }) (λ k l → hcomp (λ m → λ {
        (k = i0) → _
      ; (k = i1) → p i
      ; (l = i0) → a
      ; (l = i1) → _
      }) (p j))
```

```
λ i j k → hcomp (λ l → λ {
      (i = i0) → p j (k ∧ l)
    ; (i = i1) → p j (k ∧ l)
    ; (j = i0) → q i k
    ; (j = i1) → q i k
    ; (k = i0) → a
    ; (k = i1) → p j l
    }) (q i k)
```

```
λ i j → hcomp (λ k → λ {
    (i = i0) → _
  ; (i = i1) → _
  ; (j = i0) → a
  ; (j = i1) → λ k l → hcomp (λ m → λ
        (k = i0) → law (i ∧ l) l
      ; (k = i1) → law l (i ∧ l)
      ; (l = i0) → _
      ; (l = i1) → law k k
      }) (a )
  }) (λ k l → hcomp (λ m → λ {
        (k = i0) → p (i ∧ j)
      ; (k = i1) → r (i ∧ j)
      ; (l = i0) → a
      ; (l = i1) → _
      }) (a ))
```

In summary

n-tuples of terms in m-element bounded distributive lattice

$$\simeq$$

monotone maps $\{0 < 1\}^m \to \{0 < 1\}^n$

Formal definitions?

$\square_{\wedge\vee}$ is the full subcategory of the category of posets and monotone maps with objects $\mathbf{I}^n$ for $n \geq 0$, where $\mathbf{I} = \{0 < 1\}$.

All morphisms in $\square_{\wedge\vee}$ are of the form $\mathbf{I}^m \to \mathbf{I}^n$, e.g.:

$$s^i : \mathbf{I}^n \to \mathbf{I}^{n-1}, (e_1 \dots e_n) \mapsto (e_1 \dots e_{i-1} e_{i+1} \dots e_n) \text{ for } 1 \leq i \leq n$$

$$d^{(i,e)} : \mathbf{I}^{n-1} \to \mathbf{I}^n, (e_1 \dots e_{n-1}) \mapsto (e_1 \dots e_{i-1} e e_{i+1} \dots e_{n-1}) \quad \text{for } 1 \leq i \leq n, e \in \{0, 1\}$$

Cubical sets are objects of $\mathbf{Set}^{\square_{\wedge\vee}{}^{op}}$.

Given an n-cell p of a cubical set X and a poset map $\sigma : \mathbf{I}^m \to \mathbf{I}^n$, call $p\langle\sigma\rangle := X(\sigma)(p)$ an m-contortion of p.

TODO DO WE WANT BELOW NOTION ON BOUNDARY?
Its boundary is $\partial(p) := [p\langle d^{(1,\mathbf{0})}\rangle, p\langle d^{(1,\mathbf{1})}\rangle, \dots p\langle d^{(n,\mathbf{0})}\rangle, p\langle d^{(n,\mathbf{1})}\rangle]$

Higher inductive types

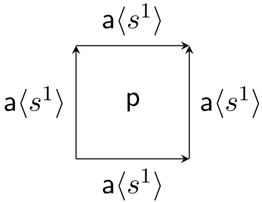
We describe cubical sets by giving the generating cells:

A <u>context</u> Γ is a list of declarations $[p_1 : T_1, \dots, p_k : T_k]$. The cubical set X generated by Γ has non-degenerate cells p_i with $\partial(p_i) = T_i$ valid boundaries and all necessary contortions.

Interval:
$[\mathsf{zero} : [], \mathsf{one} : [], \mathsf{seg} : [\mathsf{zero}, \mathsf{one}]]$

$$\mathsf{zero} \xrightarrow{\mathsf{seg}} \mathsf{one}$$

Sphere:
$[\mathsf{a} : [], \mathsf{p} : [\mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle]]$



Triangle
$[\mathsf{x} : [], \mathsf{y} : [], \mathsf{z} : [], \mathsf{p} : [\mathsf{x}, \mathsf{y}], , \mathsf{q} : [\mathsf{y}, \mathsf{z}], \mathsf{r} : [\mathsf{x}, \mathsf{z}], \phi : [\mathsf{p}, \mathsf{r}, \mathsf{x}\langle s^1\rangle, \mathsf{q}]]$

Boundaries as proof goals

Second loopspace: $[\mathsf{a} : [], \mathsf{p} : [\mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle, \mathsf{a}\langle s^1\rangle]]$

Goal: Find $\sigma : \mathbf{I}^3 \to \mathbf{I}^2$ such that $p\langle\sigma\rangle$ has this boundary:

$$[\mathsf{p}\langle\begin{smallmatrix}00\mapsto 00\\01\mapsto 01\\10\mapsto 01\\11\mapsto 11\end{smallmatrix}\rangle, \mathsf{a}\langle s^2\rangle, \mathsf{a}\langle s^2\rangle, \mathsf{a}\langle s^2\rangle, \mathsf{a}\langle s^2\rangle, \mathsf{a}\langle s^2\rangle]$$

In Cubical:

$$\mathsf{PathP}(\lambda i \to \mathsf{PathP}(\lambda j \to \mathsf{Path}\ (\mathsf{p}\ (i \wedge j)\ (i \vee j))\ \mathsf{a})$$
$$(\lambda j \to \mathsf{a})(\lambda j \to \mathsf{a}))(\lambda ij \to \mathsf{a})(\lambda ij \to \mathsf{a})$$

There are $D_3^2 = 20^2 = 400$ poset maps $\mathbf{I}^3 \to \mathbf{I}^2$ which could fit.

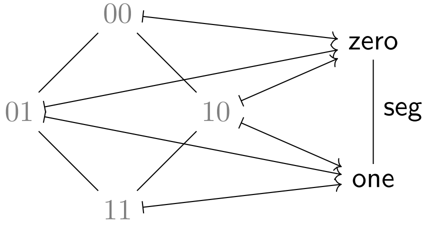
In general: D_m^n many morphisms $\mathbf{I}^m \to \mathbf{I}^n$, where D_m is the m-th Dedekind number ($D_5 = 7581, D_6 = 7828354, ..., D_9 =?$).

$\to$ Need efficient representation for collections of poset maps.

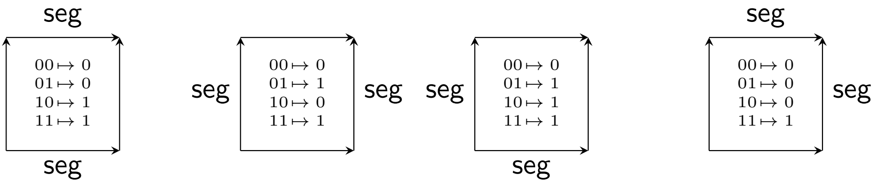
Potential contortions

ppms give us some information of all poset maps in them:

$$\mathsf{seg}\left\langle \begin{array}{l} 00 \mapsto \{0\} \\ 01 \mapsto \{0,1\} \\ 10 \mapsto \{0,1\} \\ 11 \mapsto \{1\} \end{array} \right\rangle$$



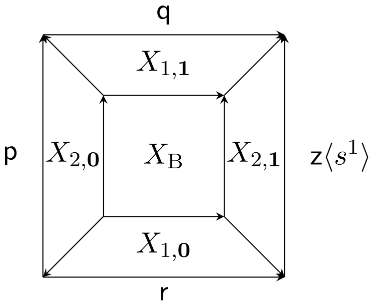
Thereby we have captured these four squares:



Triangle slide

Triangle:
$[\mathsf{x} : [], \mathsf{y} : [], \mathsf{z} : [], \mathsf{p} : [\mathsf{x}, \mathsf{y}], , \mathsf{q} : [\mathsf{y}, \mathsf{z}], \mathsf{r} : [\mathsf{x}, \mathsf{z}], \phi : [\mathsf{p}, \mathsf{r}, \mathsf{x}\langle s^1 \rangle, \mathsf{q}]]$

Goal: Find a cell with boundary $[\mathsf{r}, \mathsf{q}, \mathsf{p}, \mathsf{z}\langle s^1 \rangle]$

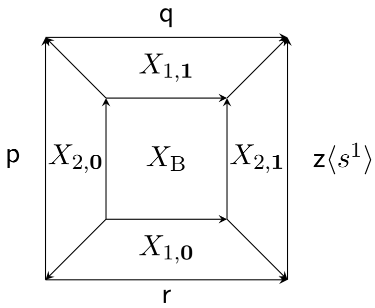


$$D_{1,0} = \{\mathsf{r}\langle \begin{array}{l} 00 \mapsto \{0\} \\ 01 \mapsto \{0,1\} \\ 10 \mapsto \{0\} \\ 11 \mapsto \{1\} \end{array} \rangle\}$$

...

$$D_B\{..., \phi\langle \begin{array}{l} 00 \mapsto \{00, 01, 10, 11\} \\ 01 \mapsto \{00, 01, 10, 11\} \\ 10 \mapsto \{00, 01, 10, 11\} \\ 11 \mapsto \{00, 01, 10, 11\} \end{array} \rangle\}$$

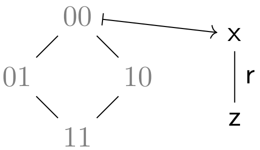
Vertex constraint

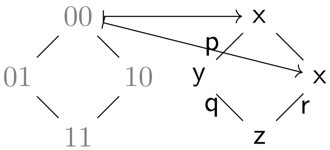


$$D_{1,0} = \{\mathsf{r}\langle \begin{array}{l} 00 \mapsto \{0\} \\ 01 \mapsto \{0,1\} \\ 10 \mapsto \{0\} \\ 11 \mapsto \{1\} \end{array} \rangle\}$$

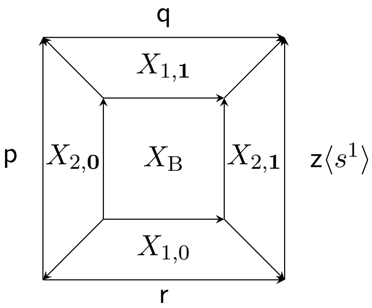
...

$$D_B = \{\ldots, \phi\langle \begin{array}{l} 00 \mapsto \{00, \cancel{01}, 10, \cancel{11}\} \\ 01 \mapsto \{00, 01, 10, 11\} \\ 10 \mapsto \{00, 01, 10, 11\} \\ 11 \mapsto \{00, 01, 10, 11\} \end{array} \rangle\}$$





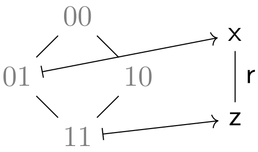
Edge constraint

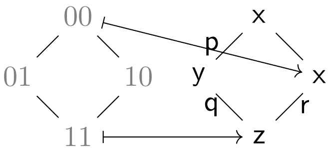


...

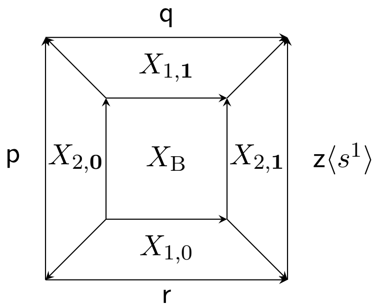
$$D_{1,0} = \{\mathsf{r}\langle \begin{array}{l} 00 \mapsto \{0\} \\ 01 \mapsto \{0\} \\ 10 \mapsto \{0\} \\ 11 \mapsto \{1\} \end{array} \rangle\}$$

$$D_{2,1} = \{\ldots, \phi\langle \begin{array}{l} 00 \mapsto \{\cancel{00}, 10\} \\ 01 \mapsto \{00, 01, 10, 11\} \\ 10 \mapsto \{11\} \\ 11 \mapsto \{11\} \end{array} \rangle\}$$





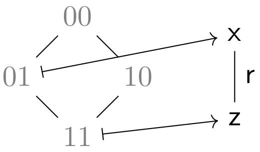
Edge constraint

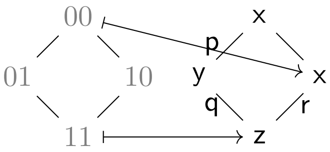


...

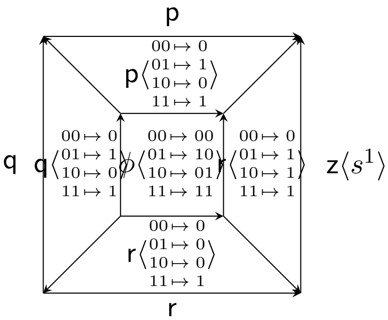
$$D_{1,0} = \{\mathsf{r}\langle \begin{matrix} 00 \mapsto \{0\} \\ 01 \mapsto \{0\} \\ 10 \mapsto \{0\} \\ 11 \mapsto \{1\} \end{matrix} \rangle\}$$

$$D_{2,1} = \{\ldots, \phi\langle \begin{matrix} 00 \mapsto \{\cancel{00}, 10\} \\ 01 \mapsto \{\cancel{00}, \cancel{01}, 10, 11\} \\ 10 \mapsto \{11\} \\ 11 \mapsto \{11\} \end{matrix} \rangle\}$$



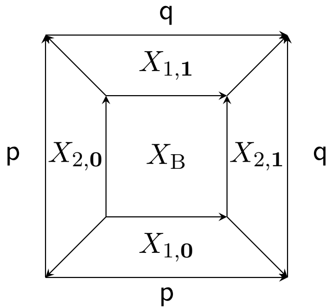


CSP solution



Example

Triangle:
$[\mathsf{x} : [], \mathsf{y} : [], \mathsf{z} : [], \mathsf{p} : [\mathsf{x}, \mathsf{y}], , \mathsf{q} : [\mathsf{y}, \mathsf{z}], \mathsf{r} : [\mathsf{x}, \mathsf{z}], \phi : [\mathsf{p}, \mathsf{r}, \mathsf{x}\langle s^1 \rangle, \mathsf{q}]]$



$$D_{1,0} = \{\mathsf{p}\langle \substack{00 \mapsto 0 \\ 01 \mapsto 0,1 \\ 10 \mapsto 0 \\ 11 \mapsto 1} \rangle\}$$

$$D_{1,1} = \{\mathsf{q}\langle \substack{00 \mapsto 0 \\ 01 \mapsto 0,1 \\ 10 \mapsto 0 \\ 11 \mapsto 1} \rangle\}$$

$$D_{1,0} = \{\mathsf{p}\langle \substack{00 \mapsto 0 \\ 01 \mapsto 0,1 \\ 10 \mapsto 0 \\ 11 \mapsto 1} \rangle\}$$

$$D_{1,1} = \{\mathsf{q}\langle \substack{00 \mapsto 0 \\ 01 \mapsto 0,1 \\ 10 \mapsto 0 \\ 11 \mapsto 1} \rangle\}$$

$$D_B = \{\mathsf{x}\langle \substack{00 \mapsto () \\ 01 \mapsto () \\ 10 \mapsto () \\ 11 \mapsto ()} \rangle, ..., \mathsf{p}\langle \substack{00 \mapsto 0,1 \\ 01 \mapsto 0,1 \\ 10 \mapsto 0,1 \\ 11 \mapsto 0,1} \rangle, ..., \phi\langle \substack{00 \mapsto 00,01,10,11 \\ 01 \mapsto 00,01,10,11 \\ 10 \mapsto 00,01,10,11 \\ 11 \mapsto 00,01,10,11} \rangle\}$$

CSP solution