Searching in cubes

Maximilian Doré

October 28, 2022

As observed by [?], do not require a full De Morgan structure. In particular, it suffices to consider only $\vee$ and $\wedge$ forming a bounded free distributive lattice.

1 Cubical sets and their boundaries

1.1 The object/universe of study

The Dedekind cube category $\square_{\wedge\vee}$ is the full subcategory of the category of posets and monotone maps with objects $\mathbf{1}^n$ for $n \geq 0$, where $\mathbf{1} = \{0 < 1\}$. Therefore morphisms in $\square_{\wedge\vee}$ are of the form $\mathbf{1}^m \to \mathbf{1}^n$. The cardinality of $\mathrm{Hom}(\mathbf{1}^m, \mathbf{1})$ is the m-th Dedekind number, which explains the name of $\square_{\wedge\vee}$.

We denote elements of $\mathbf{1}^n$ with $(e_1 \dots e_n)$ where $e_i \in \{0,1\}$ for $1 \leq i \leq n$ (we sometimes may omit the brackets). For an element $x = (e_1 \dots e_n)$ write $x_i := e_i$. The poset $\mathbf{1}^0$ has one element $()$.

All poset maps $\mathbf{1}^m \to \mathbf{1}^n$ can be obtained as compositions of the following poset maps:

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

Cubical sets are objects of $\mathbf{Set}^{\square_{\wedge\vee}{}^{op}}$, i.e., presheaves over the Dedekind cube category. For a cubical set X write $X_n := X(\mathbf{1}^n)$, which are called the n-cells of X. Given an element p of X_n, we call $\mathsf{dim}(p) := n$ the dimension of p. Write also $s_i := X(s^i) : X_{n-1} \to X_n$ and $d_{(i,e)} := X(d^{(i,e)}) : X_n \to X_{n-1}$, these maps are called *degeneracy maps* and *face maps*, respectively. Note that in any non-empty cubical set X, we must have non-empty X_n for all n: given an n-cell p, the codomain X_{n+1} of s_{n+1} as well as the codomain X_{n-1} of $d_{(n,\text{-})}$ must be non-empty. The objects in the image of some s_i are called *degeneracies*. If the dimension n is understood, we will sometimes denote with s_m the composite $s_m \circ s_{m-1} \circ \ldots \circ s_{n+1}$. This gives the shorthand $s_m(p)$ for an n-cell considered to be a degenerate m-cell.

Example 1. The cubical set Int is generated by the following data: $\mathsf{Int}_0 = \{\mathsf{zero}, \mathsf{one}\}$ and $\mathsf{seg} \in \mathsf{Int}_1$ with $d_{(1,0)}(\mathsf{seg}) = \mathsf{zero}$ and $d_{(1,1)}(\mathsf{seg}) = \mathsf{one}$. We have degenerate cells in higher dimensions for the s_i to have a codomain, for instance, $s_1(\mathsf{zero})$ is a degenerate 1-cell which is constantly zero, captured by the fact that the face maps $d_{(1,\text{-})}$ send $s_1(\mathsf{zero})$ back to zero.

Example 2. We define the cubical set $\mathsf{2Loop}$ with the following data: We have a single 0-cell, so $\mathsf{2Loop}_0 = \{\mathsf{a}\}$. There is one non-degenerate 2-cell $\alpha \in \mathsf{2Loop}_2$, which has as its faces a as a degenerate 1-cell, i.e., $d_{(2,0)}(\alpha) = d_{(2,1)}(\alpha) = d_{(1,0)}(\alpha) = d_{(1,1)}(\alpha) = s_1(\mathsf{a})$.

Example 3. We define the cubical set $\mathsf{Triangle}$ as generated by the following data: The 0-cells are $\mathsf{Triangle}_0 = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$. The non-degenerate 1-cells are $\{\mathsf{p}, \mathsf{q}, \mathsf{r}\} \subseteq \mathsf{Triangle}_1$ on which the face maps are defined by $d_{(1,0)}(\mathsf{p}) = \mathsf{x}$ and $d_{(1,1)}(\mathsf{p}) = \mathsf{y}$, $d_{(1,0)}(\mathsf{q}) = \mathsf{y}$ and $d_{(1,1)}(\mathsf{q}) = \mathsf{z}$, $d_{(1,0)}(\mathsf{r}) = \mathsf{x}$ and $d_{(1,1)}(\mathsf{r}) = \mathsf{z}$. We have one non-degenerate 2-cell $\phi \in \mathsf{Triangle}_2$ with $d_{(1,0)}(\phi) = \mathsf{p}$, $d_{(1,1)}(\phi) = \mathsf{r}$, $d_{(2,0)}(\phi) = s_1(\mathsf{x})$ and $d_{(2,1)}(\phi) = \mathsf{q}$.

Example 4. Associativity of three paths

Example 5. Given a set of generators $\{a, ...\}$ of a group G, define Group with a single 0-cell $\mathsf{Group}_0 = \{\star\}$, 1-cells $\{\mathsf{a}, \mathsf{a}^{-1}...\} \subseteq \mathsf{Group}_1$ and a 2-cell idem_a for any generator a with $d_{(2,0)}(\mathsf{idem}_\mathsf{a}) = \mathsf{a}$, $d_{(1,1)}(\mathsf{idem}_\mathsf{a}) = \mathsf{a}^{-1}$ and $d_{(2,1)}(\mathsf{idem}_\mathsf{a}) = d_{(1,0)}(\mathsf{idem}_\mathsf{a}) = s_1(\star)$.

TODO Uniform Kan condition

A type is a cubical set satisfying the uniform Kan condition.

1.2 Distortions and contexts

These examples suggest that we only require a bit of generating data from which the other cells and maps can be inferred in a unique way. This is what the next section is for.

We want to specify what a collection of face maps does???

In general, If we have an n-cell p, we can turn it into an m-cell with a map $\sigma : \mathbf{1}^m \to \mathbf{1}^n$: TODO GEOMETRIC INTUITION

This gives many cells, complex to describe (this is what is what the boundary below is for)

We will call the morphisms of $\square_{\wedge\vee}$ maps substitutions.

TODO GIVE THIS IS A NAME – NOT DEGENERACY, BUT MORE GENERAL. MAYBE contortion? OR substitution? Or substitution gives rise to contortion? OR deformation?

TODO EXAMPLE

TODO NOTATION $\begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix}$

We will call poset maps substitutions in the following (they use interval substitutions in the syntactic characterisation).

The examples suggests that we need a more concise notation to describe cubical sets. We only give the generating cells and their boundaries, which leads to the notion of higher inductive types in Cubical Agda.

In a cubical set X, the boundary of an n-cell p, called ∂p, is the union of the images of the maps $d_{(i,e)} : X_n \to X_{n-1}$, which are $2 \cdot n$ many $(n-1)$-cells. Conversely, we can specify a cubical set by declaring each cell with their boundary.

We will consider a fixed cubical set X in the following.

Definition 1. A *term* t is a tuple (p, σ) where $\sigma : \mathbf{1}^n \to \mathbf{1}^{\mathsf{dim}(p)}$. We call $\mathsf{dim}(t) := n$ the dimension of t.

TODO OR IT IS COMPOSITION OF OPEN BOX

For example, the degenerate 1-cell $s_1(\mathsf{zero})$ from Example 1 is represented in our setting by the term $(\mathsf{zero}, \begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix})$.

Definition 2. A *boundary* T is a list of tuples $[(t_1, s_1), \ldots, (t_n, s_n)]$ where each $t_\_$, $s_\_$ is a term of dimension $n-1$. We call $\mathsf{dim}(T) := n$ the dimension of T.

Definition 3. A *context* Γ is a list of declarations $[(p_1, T_1), \ldots, (p_k, T_k)]$ where each $p_\_$ is a term and each $T_\_$ is a boundary. We call $|\Gamma| := k$ the length of the context.

A cubical set is said to be generated by a context if TODO HOW EXACTLY

TODO WELL-FORMEDNESS OF CUBICAL SET DESCRIPTION

Example 1 (continued)**.** The following context generates Int: TODO

Example 2 (continued)**.** The following context generates $\mathsf{2Loop}$:
$[(\mathsf{a}, []), (\alpha, [((a, \begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix}), (a, \begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix})), ((a, \begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix}), (a, \begin{smallmatrix}0\mapsto () \\ 1\mapsto ()\end{smallmatrix}))])]$

Example 3 (continued)**.** The following context generates $\mathsf{Triangle}$: TODO

We will only consider certain boundaries. More general boundary problem is conceivable, for now we limit ourselves to the following:

We have $\partial((s_2 \circ s_1)\mathsf{a}) = \partial\alpha$.

Allowable boundary shapes are generating cofibrations from a model categorical perspective

This function computes $\partial(p)$

Algorithm 1 Computing the boundary of a term with substitution

---

Input: (p, σ) where p is an n-cell of X and $\sigma : \mathbf{1}^m \to \mathbf{1}^n$.
Output: $[faces, ...,]$

procedure BOUNDARY(p, σ)
  Do Something

For instance, in Example 2 the term $(\phi, \substack{00\mapsto 00 \\ 01\mapsto 00 \\ 10\mapsto 01 \\ 11\mapsto 01})$ picks out the first face of ϕ and is judgmentally (?) equal to $(p, \substack{00\mapsto 0 \\ 01\mapsto 0 \\ 10\mapsto 1 \\ 11\mapsto 1})$ TODO REALLY

Algorithm 2 Normalize a substituted term to normal form

Input: (p, xs) where p is an n-cell of X and $xs \subseteq \mathbf{1}^?$.
Output: (p, σ) OR ONLY σ

procedure NORMALIZE(p, xs)
  Do Something

TODO RUNTIME

Algorithm 3 Well-formed boundary

Input: Γ context, T term
Output: `OK` if T well formed, `Error` otherwise

procedure WELLFORMED(Γ, T)
  Do Something

Algorithm 4 Well-formed context

Input: Γ context
Output: `OK` if Γ well formed, `Error` otherwise

procedure WELLFORMED(Γ)
  Gradually build up? Or is it ok to have mutually defined cells?

---

2 Searching for cells

In a cubical set, we have very many distortions (?) lying around. Searching these is complicated. Here we present first approach to this.

The search problem is the following:

Definition 4. Given a cubical set X (given by a context Γ) and a boundary T (TODO over X?), the problem CubicalCell(X,T) is to find a term t such that $\partial(t) = T$.

In general, CubicalCell is undecidable.

If it can be solved by a distortion, it is clearly decidable since there are only finitely many substitutions for each cell in the context, and the search space is thereby finite. However, the search space grows super-exponentially: Suppose we want to check if there is an m-dimensional contortion of a 1-cell satisfying a goal, i.e., we are looking for a monotone map $\mathbf{1}^m \to \mathbf{1}$,. For $m = 6$, there are 7828354 substitutions that we would have to check and for 9 the exact number of possible substitutions has not even been computed yet `http://oeis.org/A00372`. Therefore we need something more clever to explore the search space.

2.1 Potential substitutions

For an effective algorithm, we turn to following notion.

We introduce potential substitution, which keep track of all the potential values of $\mathbf{1}^n$ a poset map σ might assign to an element of $x \in \mathbf{1}^m$.

They allow to represent all maps $\mathbf{1}^m \to \mathbf{1}^n$ at once. Solves memory problem. We need to explore the search space cleverly.

Definition 5. A *potential poset map* is a map $\Sigma : \mathbf{1}^m \to \mathcal{P}(\mathbf{1}^n)$.

From a potential poset map we can generate all poset maps. This has a worst-case runtime of Dedekind for the full one. For an efficient algorithm we instead consider an algorithm which extracts the first substitution from a potential substitution.

Algorithm 5 Potential substitution to substitution

Input: $\Sigma : \mathbf{1}^m \to \mathcal{P}(\mathbf{1}^n)$ potential poset map

Output: $\sigma : \mathbf{1}^m \to \mathbf{1}^n$ poset map

This has runtime TODO.

2.2 Simple solver

We want to try and see

Algorithm 6 Simple solver

Input: Γ context, $(p, S) \in Gamma$, T goal boundary
Output: σ s.t. $\partial((p,\sigma)) = T$ if there is such a σ, `Unsolvable` otherwise

```
procedure SimpleSolve(Γ, T)
    Σ ← {x ↦ 1^dim(S) | x ∈ 1^dim(T)}
    for t ← T do                                  ▷ consider T as set, remove tuples
        for x ←??? do
            Σ(x) ← {v | v ∈ Σ(x) such that ...}
            update to ensure monotonicity
    if ∃x ∈ 1^dim(T) such that Σ(x) = ∅ then
        return Unsolvable
    else
        σ = Extract first from Σ
        return (p, fstΣ)
```

We can try this for all elements in the context, thereby have run time $\mathcal{O}(|\Gamma|)$

3 Composition solver

We use Kan composition

4 (Un)decidability

Show how in general undecidable: Reduce to word problem

Find how decidable groups carry over to decidable problems?

5 Negation

How to encode search for reversals

6 Connections

Since the generating data is the non-degenerate cells and the face maps on them, we can adopt the following notation to describe cubical sets, which go under higher inductive types in Cubical Agda:

TODO also boundary description with poset maps. These correspond to specifying the dmaps: One index is constant, rest is given by given poset map

6.0.1 Relation to face formulas

The morphisms in $\square_{\wedge\vee}$ have a succinct description as tuples of elements of a free distributive lattices, which we will call telescopes. Cubical Agda uses telescopes to describe this. However, these are hard to construct and have no geometric intuition. This is why we had used poset maps. We can go back and forth between both representations easily.

Given $s : \mathbf{1}^m \to \mathbf{1}^n$, we compute an n-tuple of elements of the free distributive lattice over m elements as follows:

The i-th entry is $\{x \in \mathbf{1}^m \mid s(x)_i = 1\}$. An element x can be seen as a clause if we regard it as indicator of which elements of the lattice are used, e.g., $(1, 0, 1)$ represents the clause $x \vee z$ if the three elements of the lattice are x, y and z.

Algorithm 7 TODO

Input: $\sigma : \mathbf{1}^m \to \mathbf{1}^n$

Output: ϕ n-tuple of elements of free distributive lattice over m variables

```
procedure SUBST2TELE(σ)
    for i ← 1 to n do
        φ_i ← {x ∈ 1^m | σ(x)_i = 1}        ▷ O(2^m) many elements in 1^m
    return (φ_1, ..., φ_n)
```

TODO mention set representation of DNF. Also mention normalization necessary – what's its runtime?

From an n-tuple of formulas over ϕ we can read off a poset map $s : \mathbf{1}^m \to \mathbf{1}^n$ as follows: Given an element $x \in \mathbf{1}^m$, $s(x) = (e_1, ..., e_n)$ where $e_i = \phi_i@x$ TODO EVALUATION

Algorithm 8 TODO

Input: TODO

Output: TODO

```
procedure Tele2Subst(p)
    for x ← 1^m do
        for i ← 1 to n do
            σ(x)_i ← φ_i@x
    return σ
```

Therefore going back and forth between formulas and poset map takes $\mathcal{O}(2^m n)$ many steps. This is linear in the number of elements of the data structures we are considering, so pretty quick and no obstruction.

Example 6.

TODO PROOF THAT THESE ARE MUTUALLY INVERSE?

6.1 Nominal perspective

Have set $I = \{i, j, k, ...\}$ of names. Then give complete description for Cubical Agda.