plan for my classes

• 18/09
  • categories, monos, push-outs
  • expansive rewriting
  • forward propagation
  • homework assignment #1

• 2/10
  • pull-backs, pull-back complements
  • restrictive rewriting
  • backward propagation
  • homework exercise #2

• 16/10
  • graph-based knowledge representation
  • the KAMI bio-curation system
  • choice of research papers to present
expansive rewriting
intuitively

• to rewrite a graph $G$, need a rule $r : L \rightarrow R$, and a matching $m : L \rightarrow G$
  • intuition [from last week]: replace the image (via $m$) of $L$ in $G$ by $R$
  • $R$ may contain nodes/edges that are not in the image (via $r$) of $L$: these are added to $G$
  • $R$ may contain nodes that are the image (via $r$) of multiple nodes in $L$: these are merged in $G$
• if the graph $G$ is typed, i.e. we have $h : G \rightarrow T$, these changes may propagate
  • we must specify which added nodes/edges can already be typed in $T$ [requires a choice]
  • if nodes of different types are merged, their types must also be merged [canonical]
• we are seeking an abstract means to express these aspects of rewriting
  • uses the language of category theory…
categories

- a **pre-order** on a set $S$ is a binary relation $R$ on $S$ such that
  - reflexive: for all $s \in S$, $s R s$
  - transitive: for all $s_1, s_2, s_3 \in S$, if $s_1 R s_2$ and $s_2 R s_3$ then $s_1 R s_3$
    - can also be seen as a kind of simple directed graph: where there’s a path, there’s an edge
  - $s_1, s_2 \in S$ are **isomorphic**, written $s_1 \cong s_2$, iff $s_1 R s_2$ and $s_2 R s_1$
    - quotienting by $\cong$ yields a partial order, i.e. antisymmetric: for all $s_1, s_2 \in S$, if $s_1 R s_2$ and $s_2 R s_1$ then $s_1 = s_2$

- a **category** is a collection of objects and a collection of arrows
  - let $A, B, \ldots$ range over objects and $f, g, \ldots$ range over arrows
  - an arrow has a source and a target object, i.e. $f : A \to B$

where
- each object $A$ has an **identity** arrow $1_A : A \to A$
- each pair of arrows $f : A \to B$ and $g : B \to C$ [target of $f =$ source of $g$] has a **composite** $g \circ f : A \to C$

satisfying
- identity: for all $f : A \to B$, $f \circ 1_A = f = 1_B \circ f$
- associativity: for all $f : A \to B$, $g : B \to C$ and $h : C \to D$, $h \circ (g \circ f) = (h \circ g) \circ f$

- some examples
  - sets and functions, groups and their homomorphisms, vector spaces and linear maps, …
  - (simple) graphs and their homomorphisms
    - why does $g \circ f$ preserve edges?
arrows

• some special kinds of arrows
  • \( f : A \rightarrow B \) is an **isomorphism** iff, for some \( g : B \rightarrow A \), \( g \cdot f = 1_A \) and \( f \cdot g = 1_B \)
  • \( f : A \rightarrow B \) is a **monomorphism** iff, for any pair \( g,h : X \rightarrow A \) of arrows, if \( f \cdot g = f \cdot h \) then \( g = h \)
    • in words, \( f \) makes no identifications
    • alternatively, \( f \) is post-cancellable
    • to assert that \( f \) is a mono(morphism), we write \( f : A \rightarrowtail B \)
  • in the category **Set** of sets and functions, the monos are precisely the injective functions
    • the notion of inclusion / subset
  • monos in **Grph**, the category of graphs and homomorphisms, are used in graph rewriting
    • the notion of **matching**

• quick exercises
  • if \( f : A \rightarrow B \) is an isomorphism, the arrow \( g \) is unique [and is usually written \( f^{-1} \)]
  • if \( f : A \rightarrowtail B \) and \( g : B \rightarrowtail C \) then \( g \cdot f : A \rightarrowtail C \) [monos are closed under composition of arrows]
  • suppose \( f : A \rightarrow B \), \( g : B \rightarrowtail C \) and \( g \cdot f : A \rightarrowtail C \)
    • show that \( f : A \rightarrowtail B \)
    • give an example where \( g \) is not a mono

• sub-categories
  • sub-collections of objects and arrows closed under composition
  • **Set** is the category of sets and injective functions
given a category $C$ and an object $T$, the slice category $C / T$ is defined as
- objects: all arrows $f : X \rightarrow T$ of $C$
- given objects $f : X \rightarrow T$ and $g : Y \rightarrow T$, an arrow from $f$ to $g$ is an arrow $h : X \rightarrow Y$ of $C$ satisfying $f = g \cdot h$
- in words, objects are arrows and arrows are commuting triangles

what is $\text{Set} / T$?

what is $\text{Set}^\triangleright / U$?
categorical constructions

- many familiar constructions on sets can be defined purely in terms of categories
  - no need to refer to ‘sets’ or ‘elements’ or ‘functions’, &c.
  - just refers to (objects and) arrows and the properties that we require [the concept of mono is a simple example]
  - this use of category theory abstracts away from the specific details of the objects and arrows
    - a way to identify expected, or unexpected, commonalities between different mathematical theories
    - also a labour-saving technique: a result that can be proved purely categorically can easily be transferred to many specific concrete settings
- very often, and always in this class, this works as follows
  - we have some (given) starting data: objects and arrows
  - we state a construction: some more objects and arrows [that we want to exist]
  - we state the universal property that the construction must satisfy: a specification of the construction
- easiest to understand with an example!
  - given two objects $A$ and $B$
  - we ask that there exists an object $C$ and two arrows $i_A : A \to C$ and $i_B : B \to C$
  - satisfying: for any (other) object $D$ and arrows $f : A \to D$ and $g : B \to D$, there exists a unique arrow $h : C \to D$
    such that $f = h \cdot i_A$ and $g = h \cdot i_B$

- what are $C$ and the arrows $i_A : A \to C$ and $i_B : B \to C$?
  - in the category of sets and inclusions
  - in $\text{Set}$
  - in category theory, this construction is called a co-product
push-outs

generalized co-products

• in the category of sets and inclusions, i.e. $\text{Set}^\rightarrow / U$
  • if $A \subseteq B$ and $A \subseteq C$ then $D := B \cup C$ is (still) the smallest set such that $B \subseteq D$ and $C \subseteq D$
  • same as on the previous slide [where $A = \emptyset$]
• in the category of sets and monos, i.e. $\text{Set}^>$
  • if $f : A \rightarrow B$ and $g : A \rightarrow C$ then $D := A + (B - A) + (C - A)$ is the smallest set with $g' : B \rightarrow D$ and $f' : C \rightarrow D$ such that $g' \circ f = f' \circ g$
  • varying the set $A$ varies the result $D$ — unlike in $\text{Set}^> / U$ [which is a rather strange special case; why?]
    • if $A = \emptyset$, we get $D = B + C$
    • as $A$ gets bigger, we quotient $B + C$ by identifying pairs of elements

• the general notion of push-out
  • given a pair of arrows $f : A \rightarrow B$ and $g : A \rightarrow C$
  • we want an object $D$ and a pair of arrows $g' : B \rightarrow D$ and $f' : C \rightarrow D$ such that $g' \circ f = f' \circ g$
  • satisfying: for any (other) object $E$ and arrows $g'' : B \rightarrow E$ and $f'' : C \rightarrow E$, there exists a unique arrow $h : D \rightarrow E$ such that $g'' = h \circ g'$ and $f'' = h \circ f'$

• what happens in $\text{Set}$? and in $\text{Grph}$?
expansive rewriting
formally

- assume a category with all push-outs
  - or at least “push-outs over monos”
- a rule is an arrow $r : L \rightarrow R$ and a matching is a mono $m : L \rightarrow G$
  - take the push-out of $r$ and $m$: $m^+ : R \rightarrow G^+$ and $r^+ : G \rightarrow G^+$
  - $G^+$ is the updated version of $G$
  - $m^+$ is a matching of the RHS of $r$ into $G^+$
  - $r^+$ is the instantiation of $r$ to $G$

- we have seen that $\textbf{Set}$ and $\textbf{Grph}$ have all push-outs
  - intuitive that $\textbf{Set} / T$ and $\textbf{Grph} / T$ also do
  - but how do we prove this categorically [i.e. for all slice categories]?
two technical points

• the **pasting lemma** for push-outs
  • if the two inner squares are push-outs then so is the outer rectangle

  \[
  \begin{array}{ccc}
  \text{A} & \rightarrow & \text{C} & \rightarrow & \text{E} \\
  \downarrow & & \downarrow & & \downarrow \\
  \text{B} & \rightarrow & \text{D} & \rightarrow & \text{F}
  \end{array}
  \]

  • proof?

• the **image factorization** of an arrow \(f : A \rightarrow B\)
  • an object \(I\) and an arrow \(m : I \rightarrow B\) such that
    • there exists an arrow \(e : A \rightarrow I\) such that \(f = m \cdot e\)
    • for any other \(e' : A \rightarrow I'\) and \(m' : I' \rightarrow B\) such that \(f = m' \cdot e'\), there is a unique \(i : I \rightarrow I'\) such that \(m = m' \cdot i'\)
  • in words, \(I\) is the **smallest** object through which we can factorize \(f\) with a mono \(I \rightarrow B\)
propagation

\[ G \rightarrow T \]

- suppose we have a rule \( r : L \rightarrow R \), a matching \( m : L \rightarrow G \) and a typing \( h : G \rightarrow T \)
  - we want to apply \( r \) to \( G \) via \( m \) — but we may not be able to type all of these changes in \( T \)
  - we split \( r \) into two phases:
    - the strict phase, which makes all changes to \( G \) that can be typed by \( T \)
    - the propagation phase, which performs the remaining changes to \( G \) and propagates these changes to \( T \)

\[ L \xrightarrow{r} R \]

\[ h \cdot m \]

\[ T \leftarrow S \]

- let us note that
  - by the pasting lemma, the two-stage rewrite of \( G \) is equivalent to using \( r \) directly
  - the propagation phase is not, in general, a rule application
    - the 'matching' is not a mono!
    - we can construct a bona fide rule application using the image factorization of the arrow \( S \rightarrow T \)
propagation

general case

• if we have multiple levels of typing, e.g. \( G \rightarrow T \rightarrow U \)
  • given a rule to be applied to \( G \), we need one factorization for \( T \) and another for \( U \)
  • these factorizations must be compatible: the strict phase for \( U \) includes that for \( T \)

• if we have branching, e.g. \( T \leftarrow G \rightarrow U \)
  • we still need one factorization for \( T \) and another for \( U \)
  • no compatibility is needed at this level — but will be required later if we “close the diamond”

• in general
  • all graphs below \( G \) are susceptible to be modified by propagation
  • update sink graphs first; then the pre-sinks; &c.
    • this allows to perform \textbf{in-place} update
    • more efficient, at least for in-memory implementations
  • our implementation in Neo4j proceeds in the other direction
    • breaks typing at each rewrite
    • but we can exploit the QL to \textbf{repair} locally — and this performs propagation