Solutions set for Exercise Group #2

CAS 701 Logic and Discrete Mathematics

Musa Al-hassy

« Fall 2017 »

Abstract

Induction and Orders

Students were required to submit solutions to 10 of the given 15 exercises posed by professor William Farmer.
Notationally, I use associative equivalence ' ≡ ' in-place of conjunctive ' $\iff$ '.
This document renders its calculations using Maarten Fokkinga's calculation.sty LaTeX style file.

Contents

0 Preliminaries

0.1 Quantification

In programming we write unambiguous code rather than phrases such as "and so on" or "...". Indeed, for such situations we use loops.

In the same vein, we write

$$(\oplus i \mid 0 \le i < n \bullet f\, i) \;=\; f\,0 \oplus f\,1 \oplus \cdots \oplus f\,(n-1)$$

Formally, we can define this by recursion on n.

This notation is similar to the `for-loop` setup of imperative programming,

$$(\oplus \text{ variable declarations} \mid \text{constraints on vars} \bullet \text{body})$$

Recursion-induction would define the notation for finite ranges only, we avoid that by axiomatisating the notation directly:

- Empty-range: $(\oplus\ i \mid \mathit{false} \bullet f\ i) = e$, where e is the identity of $\oplus$.
- One-point rule: $(\oplus\ i \mid i = k \bullet f\ i) = f\ k$.
- Split-off term: $(\oplus\ i : \mathbb{Z} \mid i < n{+}1 \bullet f\ i) = (\oplus\ i : \mathbb{Z} \mid i < n \bullet f\ i) \oplus f\ n$.
- Distributivity: $(\oplus\ i \mid R\ i \bullet f\ i \oplus g\ i) = (\oplus\ i \mid R\ i \bullet f\ i) \oplus (\oplus\ i \mid R\ i \bullet g\ i)$.
- Abbreviation: $(\oplus\ i \bullet f\ i) = (\oplus\ i \mid \mathit{true} \bullet f\ i)$.

Quantification underlies the theory of loops in programming, for example the distributivity rule corresponds to loop fusion/fission rules of compiler optimisation. This notation is sometimes known as Eindhoven quantifier notation. More laws for this notation can be found at `http://www.cas.mcmaster.ca/~alhassm/ThmList.pdf`

0.1.1 comparision to traditional notation

An existing two-dimensional notation for quantification that is more commonly used has some troubles. For example, $(+\ n : \mathbb{N} \mid 0 \le i \le n \bullet f\ i)$ is rendered as $\sum_{i=0}^{n} f(i)$. Some troubles include,

- As the summation example shows, a major drawback of the common notation is that it is limited to a consecutive sequence of numbers. Problems arise if the notation is to be used for quantification's over non-consecutive numbers. A hack to this problem is to write $\sum_{R(i)} f(i)$ but this is still not sufficient: What does $\sum_{i+j+m=n} f(i,j,m,n)$ mean? Is it a sum over dummy variable *i, j, m,* or n?
- The constraint is written subscript, thereby treated as a second-class citizen in the notation, and thus leading to the perception that it is less important. Moreover, the subscript and positioning make it more difficult to manipulate and so rules regarding it are less known.
- Another problem with placing the constraint as a subscript is that it does not scale-up. If the constraint gets too complicated, the two-dimensional notation becomes visually messy and even more difficult to work with when the constraint itself contains an instance of a two-dimensional quantification. This is usually avoided by making local definitions. The linear notation can handle such matters directly and immediately.
- Traditional quantifier notation uses a unique symbol, usually an enlarged variant of the underlying monoid operation, for each kind of quantification. This linear style is more uniform and makes clear the commonality of the general pattern more explicitly: Dummy variables, their ranges, and the terms used. For example, universal quantification is traditionally denoted ' $\forall\ i \bullet R\ i \Rightarrow P\ i$ ' which is radically different from that for products ' $\Pi_{i,R(i)} P(i)$ '. It is such difference that obscure the commonalities in the properties of quantifiers.
- Since the notation is systematic, the calculational rules need be given just once for a great variety of different quantifiers; resulting in a substantial reduction in the number of laws one has to remember.
- Besides an imposition of consistency, this notation makes explicit what is implicit in traditional notation: Bounds variables and their scope.

0.2 Lattices

I shall denote an arbitrary order by the not-so-familiar notation '$\sqsubseteq$' so that we do not *accidently* assume it behaves similar to the order of numbers '$\leq$'.

A *Lattice* is an algebraic structure $(L, \sqcup, \sqcap)$ where the two operations are associative, idempotent, symmetric and mutually absorptive. Alternatively, a lattice is an ordered set $(L, \sqsubseteq)$ endowed with a pair of operations $(\sqcup, \sqcap)$ that are uniquely characterised by the following *specfications*:

- Join-characterisation: $x \sqcup y \sqsubseteq z \quad \equiv \quad x \sqsubseteq z \quad \wedge \quad y \sqsubseteq z$
  - The '$\Leftarrow$' part of '$\equiv$' reads: *if z is an upper bound, then it is at-most the join.*
  - The converse yields that the join is itself an upper bound.
  - Together this specification ensures that the join operation produces the least upper bound.
- Meet-characterisation: $z \sqsubseteq x \sqcap y \quad \equiv \quad z \sqsubseteq x \quad \wedge \quad z \sqsubseteq y$

Heuristic: To find out whether an order is a lattice one may 'solve' the characterisation laws: Rewrite $x \sqsubseteq z \quad \wedge \quad y \sqsubseteq z$ using equivalence-preserving transformations to obtain an expression of the shape $w \sqsubseteq z$, then necessarily $x \sqcup y = w$.

- This is better than giving a definition of join *then* verifying that the definition is valid.
- Examples of this heuristic can be found in questions 5 and 7.

For quantified expressions, the specifications take the following forms:

- Generalised Join-characterisation: $\bigsqcup S \sqsubseteq z \quad \equiv \quad (\forall s : S \bullet s \sqsubseteq z)$
  - Taking $S = \varnothing$ makes the forall vacuously true and so we have: forall z that $\bigsqcup \varnothing \sqsubseteq z$. That is, $\bigsqcup \varnothing$ is the minimal element of the order, if it exists at all. It is usually denoted '$\bot$', *bottom*, or '$-\infty$'.
- Generalised Meet-characterisation: $z \sqsubseteq \bigsqcap S \quad \equiv \quad (\forall s : S \bullet z \sqsubseteq s)$
  - *Top* is the maximal element, if it exists: $\top = \bigsqcap \varnothing$.

Principle of duality: If P is a theorem stated in-terms of lattice operators then P' is also a theorem where P' is P with symbols $(\sqcup, \sqcap, \bot, \top)$ replaced by $(\sqcap, \sqcup, \top, \bot)$.

1 Basics of order theory

Let $\mathcal{P}$ be be the poset with order usual set-containment and carrier-set the following family:

{1} , {2}, {4}, {1,2}, {1,4}, {2,4}, {3,4}, {1, 3, 4}, {2,3,4}

- A **maximal element** is one that does not have any elements above it.
  - {1,2} , {1, 3, 4} , {2, 3, 4}
- A **minimal element** is one that does not have any elements below it.
  - {1}, {2}, {4}
- The **maximal element**, also known as a *top element*, is above all elements.
  - $\mathcal{P}$ has none.
  - In usual set-based orders, it is the universal set.
- The **minimum element**, also known as a *bottom element*, is below all elements.
  - $\mathcal{P}$ has none.
  - In usual set-based orders, it is the empty set.
- The **upper bounds** of a collection are elements that are above the collection's members.
  - The upper bounds of { {2}, {4} } are the sets that contain {2} and {4}; i.e., {2,4} and {2,3,4}.
- The **least upper bound**, or *join*, of a collection is the minimum element of its upper bounds, if it exists.
  - The join of { {2}, {4} } is {2,4}.
- The **lower bounds** of a collection are elements that are below the collection's members.
  - The lower bounds of { {1,3,4}, {2,3,4} } are the elements of $\mathcal{P}$ that are below {1,3,4} and {2,3,4}; i.e., {4} and {3,4}.
- The **least lower bound**, or *meet*, of a collection is the maximum element of its lower bounds, if it exists.
  - The meet of { {1,3,4}, {2,3,4} } is {3,4}.

2 Orders are closed under products

The product of orders, with component-wise ordering, is again an order.

- Indeed, let I be some index set for posets $\mathcal{P}_i$, where $i : I$.
- Furnish the type $\mathcal{P} := \Pi_i \mathcal{P}_i$ of 'tuples' $x = (x_i \mid i : I, x_i \in \mathcal{P}_i)$ with the the 'component-wise' relation: $x \leq y :\equiv (\forall i : I \bullet x_i \leq_i y_i)$.

It remains to show that this relation is indeed an order.

- **Reflexive** For any $x \in \mathcal{P}$,

$$
\begin{aligned}
& x \leq x \\
= \quad & \{ \text{ Definition } \} \\
& \forall i : I \bullet x_i \leq_i x_i \\
= \quad & \{ \ \mathcal{P}_i \text{ is a poset, whence reflexive } \} \\
& \forall i : I \bullet \mathsf{true} \\
= \quad & \{ \text{ Superfluous quantification } \} \\
& \mathsf{true}
\end{aligned}
$$

- **Transitive** For any $x,\ y,\ z \in \mathcal{P}$,

$$
\begin{array}{ll}
& x \leq y \quad \wedge \quad y \leq z \\
= & \quad \{ \text{ Definition } \} \\
& (\forall i : I \bullet x_i \leq_i y_i) \quad \wedge \quad (\forall i : I \bullet y_i \leq_i z_i) \\
= & \quad \{ \text{ Quantifiers } \} \\
& \forall i : I \bullet x_i \leq_i y_i \wedge y_i \leq_i z_i \\
\Rightarrow & \quad \{ \ \forall\text{-monotonicity with } \mathcal{P}_i \text{ is a poset whence transitive } \} \\
& \forall i : I \bullet x_i \leq_i z_i \\
= & \quad \{ \text{ Definition } \} \\
& x \leq z
\end{array}
$$

- **Antisymmetric** For any $x,\ y \in \mathcal{P}$,

$$
\begin{array}{ll}
& x \leq y \quad \wedge \quad y \leq x \\
= & \quad \{ \text{ Definition } \} \\
& (\forall i : I \bullet x_i \leq_i y_i) \quad \wedge \quad (\forall i : I \bullet y_i \leq_i x_i) \\
= & \quad \{ \text{ Quantifiers } \} \\
& \forall i : I \bullet x_i \leq_i y_i \wedge y_i \leq_i x_i \\
= & \quad \{ \ \mathcal{P}_i \text{ is a poset, whence antisymmetric } \} \\
& \forall i : I \bullet x_i \approx y_i \\
= & \quad \{ \text{ Component-wise equality } \} \\
& x \approx y
\end{array}
$$

3 Existentially complete spaces are invariant under bottom/top extensions

Recall, a space is *existentially complete* precisely when non-empty subsets with upper bounds actually have least upper bounds.

- The qualifier *existential* refers to the existence of an element in the mentioned subset.
- The real numbers are existentially complete and this fact is so useful that the property's name is usually elided to 'complete' —with the non-emptiness proviso being implicitly understood.

Anyhow,

- Suppose $(\mathcal{P},\ \sqsubseteq)$ is an existentially complete and dense poset.
- Let its "top extension" be $\mathcal{P}_\top := \mathcal{P} \cup \{\top\}$ where $\top \notin \mathcal{P}$ is an arbitrary new symbol.
- Define an order $- \sqsubseteq' -$ on the extended space as follows:
  - $x \sqsubseteq' y \ \equiv \ x \sqsubseteq y$ provided both $x,\ y \in \mathcal{P}$
  - $x \sqsubseteq' \top \ \equiv \ \mathsf{true}$ for any $x \in \mathcal{P}$.
- It is a simple matter to show that this relation is indeed an order.

To show that the extended space is existentially complete, let S be any non-empty subset of the extended space. Then either it contains the new element or it does not:

- If $\top \in S$ then it is a simple matter to see that $\sqcup\ S = \top$.

- If $T \notin S$ then $\sqcup\, S$ already exists in $\mathcal{P}$ and since the extended order actually extends the original order —c.f. the first clause in its definition— we have that $\sqcup\, S$ is also the join of S in the extended space.

By duality, we find that the "bottom extension" $\mathcal{P}_\perp := \mathcal{P} \cup \{\perp\}$ to be existentially meet-complete.

Consequently, the bottom-top extension $\mathcal{P}_{\perp,\top} := \mathcal{P} \cup \{\perp, \top\}$ is (existentially meet- and join-)complete. The non-emptiness proviso is lifted since the join of the empty set is the new bottom-element and the meet of the empty set is the new top-element.

- In-particular, the real numbers have $\mathbb{R} \cup \{-\infty, +\infty\}$ as their bottom-top extension, a complete ordered space.
- Recall that a space is *dense*, or a 'continuum', precisely when distinct elements necessarily have

other elements between them: $\forall\ x,y \bullet x \sqsubset y \Rightarrow \exists\ z \bullet x \sqsubset z \sqsubset y$.

- To see that the bottom-up extension of the reals is also dense, suppose $x \sqsubset y$. By definition-chasing, there are three cases:
  - $x = -\infty$: Let $z = y - 1$, which clearly satisfies the desired property.
  - $x \neq -\infty$, $y \neq +\infty$: Then both x,y are real with $x < y$ and so $\exists\ z \bullet x \sqsubset z \sqsubset y$ since the reals are complete.
  - $y = +\infty$: Let $z = x + 1$, which clearly satisfies the desired property.

Challenge: Generalise the density argument in a fashion similar to how the completness issue for $\mathbb{R} \cup \{-\infty, +\infty\}$ was a consequence of a construction.

4 The product of well-orders with the lexicographical order is again a well-order

This result is proved as *Lemma 2.1.3* on page 10 of Gallier.

5 Linear orders are lattices.

Assuming linearity, we shall *calculate* a definition for join:

$$
\begin{array}{ll}
 & x \sqsubseteq z \ \wedge\ y \sqsubseteq z \\
= & \quad \{\ \text{Linearity assumption}\ \} \\
 & (x \sqsubseteq z \ \wedge\ y \sqsubseteq z) \quad \wedge \quad (x \sqsubseteq y \ \vee\ y \sqsubseteq x) \\
= & \quad \{\ \text{Distributivity of conjunction over disjunction}\ \} \\
 & (x \sqsubseteq z \ \wedge\ y \sqsubseteq z \ \wedge\ x \sqsubseteq y) \quad \vee \quad (x \sqsubseteq z \ \wedge\ y \sqsubseteq z \ \wedge\ y \sqsubseteq x) \\
= & \quad \{\ \text{Just as } x \leq y \ \equiv\ x \,\mathsf{min}\, y = x \text{ for numbers,} \\
 & \quad\ \ \text{we have } p \Rightarrow q \ \equiv\ p \wedge q \ \equiv\ p \text{ for Booleans.} \\
 & \quad\ \ \text{We shall use the latter property with} \\
 & \quad\ \ \text{the transitvity of order: } a \sqsubseteq b \ \wedge\ b \sqsubseteq c \ \Rightarrow\ a \sqsubseteq c.\ \} \\
 & (x \sqsubseteq y \ \wedge\ y \sqsubseteq z) \quad \vee \quad (y \sqsubseteq x \ \wedge\ x \sqsubseteq z) \\
= & \quad \{\ \text{In a linear order, we have that } a \sqsubseteq b \ \equiv\ \neg(b \sqsubset a)\ \} \\
 & (x \sqsubseteq y \ \wedge\ y \sqsubseteq z) \quad \vee \quad (\neg x \sqsubset y \ \wedge\ x \sqsubseteq z) \\
= & \quad \{\ \text{Definition of conditional expression}\ \} \\
 & \mathsf{if}\ \ x \sqsubseteq y\ \ \mathsf{then}\ \ y \sqsubseteq z\ \ \mathsf{else}\ \ x \sqsubseteq z\ \ \mathsf{fi} \\
= & \quad \{\ \text{Conditionals distribute/factor out of expressions}\ \} \\
 & \mathsf{if}\ x \sqsubseteq y\ \mathsf{then}\ y\ \mathsf{else}\ x\ \mathsf{fi} \quad \sqsubseteq \quad z \\
= & \quad \{\ \textbf{Define}\ x \sqcup y = \mathsf{if}\ x \sqsubseteq y\ \mathsf{then}\ y\ \mathsf{else}\ x\ \mathsf{fi}\ \} \\
 & x \sqcup y \sqsubseteq z
\end{array}
$$

Hence, we have found a *correct-by-construction* definition of join!
By duality —i.e., replacing '$\sqsubseteq$' with '$\sqsupseteq$'—, we find that $x \sqcap y = \textsf{if } x \sqsupseteq y \textsf{ then } y \textsf{ else } x \textsf{ fi}$.

Challenge: Prove that the following are equivalent formulations of linearity.

1. "Linearity": $x \sqsubseteq y \vee y \sqsubseteq x$
2. "Join co-characterisation": $\forall z \bullet \quad z \sqsubseteq x \sqcup y \quad \equiv \quad z \sqsubseteq x \vee z \sqsubseteq y$
3. "Order Negation": $\neg(x \sqsubseteq y) \quad \equiv \quad y \sqsubset x$
   - Recall "Definition of strict order": $a \sqsubset b \equiv a \sqsubseteq b \wedge a \neq b$.
4. "Anti-Symmetry as Equivalence": $x \sqsubseteq y \quad \equiv \quad y \sqsubseteq x \quad \equiv \quad x = y$
   - Sometime this is known as 'trichotomy'.

6 The naturals under division form a complete lattice

The gcd is by definition the greatest common divisor of its arguments, that is, it is the greatest number that divides both of its arguments, and so by definition, it is the meet of its arguments when the order is divisbility.

The Euclidean Algorithm is method to compute the gcd of any pair of numbers and so it can be applied iteratively to compute the gcd of any finite collection of numbers —including the empty collection, in which we simply obtain 1. Ignoring physical limitations, the same computations could be carried out for any infinite collection of natural numbers —it would simply require countably infinte time.

In the same vein, the lowest common multiple of any collection of naturals is their join.

7 All finite non-empty joins exist in a lattice

We wish to assign meaning to $(\sqcup i : 1..n \bullet x_i)$ such that the result is in-fact the join of the set $\{x_i | i \in 1..n\}$.

Rather than giving a definition *then*, after the fact, checking that it actually satisfies our intended specification, we shall *calculate the definition of that expression with the specification guiding the calculation.*

We shall do so by induction on $n \geq 1$.

Base case

$$
\begin{array}{ll}
 & \forall i : 1..1 \bullet x_i \sqsubseteq z \\
= & \quad \{ \text{ One-point rule: There is only one-possible } i \text{ value. } \} \\
 & x_1 \sqsubseteq z
\end{array}
$$

Inductive case

$$
\begin{array}{ll}
 & \forall i : 1..n+1 \bullet x_i \sqsubseteq z \\
= & \quad \{ \text{ Split-off term } \} \\
 & (\forall i : 1..n \bullet x_i \sqsubseteq z) \quad \wedge \quad x_{n+1} \sqsubseteq z \\
= & \quad \{ \text{ Induction hypothesis } \} \\
 & (\sqcup i : 1..n \bullet x_i) \sqsubseteq z \quad \wedge \quad x_{n+1} \sqsubseteq z \\
= & \quad \{ \text{ Binary join Characterisation } \} \\
 & (\sqcup i : 1..n \bullet x_i) \sqcup x_{n+1} \sqsubseteq z
\end{array}
$$

Hence, by our induction argument we have a *correct-by-construction* definition of quantified join:

- Singleton collection: $(\sqcup i : 1..1 \bullet x_i) = x_1$
- Collection of more than one element: $(\sqcup i : 1..n+1 \bullet x_i) = (\sqcup i : 1..n \bullet x_i) \sqcup x_{n+1}$.

Where have we seen exceedingly similar formulae before? *;)*

8 Subsets form a Boolean Algebra

Consider the collection of terms formed by the following grammar,

τ := $\varnothing$ | $\tau \cup \tau$ | $\tau \cap \tau$ | ~ τ | $\mathcal{U}$ | X where X is any subset of the universal set $\mathcal{U}$

We can inductively define a map that reifies each set-theortic term as an expression in the language of Boolean logic $\mathbb{B}$ = {true, false}. Let x be an arbitrary element in the universe, then define

```
(-)_x : Set-terms → 𝔹-terms
∅_x          =  false
𝒰_x          =  true
X_x          =  x ∈ X
(X ∪ Y)_x    =  X_x  ∨ Y_x
(X ∪ Y)_x    =  X_x  ∧ Y_x
(~ X)_x      = ¬ X_x            -- complementation
```

Now the extensionality principle for set equality says that $X = Y$ precisely when $=\forall$ x • $X_x = Y_x =$. By induction it can be readily seen that for two *set terms* we have $E = F$ precisely when $\forall\ x \bullet E_x = F_x$. Now we turn to proving that the above grammar furnishes the powerset of $\mathcal{U}$ with a Boolean Algebra by proving all axioms hold.

- All Boolean Algebra axioms are equations, so let $E = F$ be such an equation whose variables are sets.
- By the extended extensionality principle, we know that this holds precisely when $\forall\ x \bullet E_x = F_x$.
- The Booleans $\mathbb{B}$ = *{true, false}* are, needless to say, a Boolean Algebra and so the Boolean expression $E_x = F_x$ holds.
- Hence, the original equation also holds in the context of sets.

The family of subsets forms a Boolean Algebra by deferring the 'heavy lifting' to the binary Boolean Algebra $\mathbb{B}$. Indeed it is the membership predicate that is responsible for this transformation.

9 The finite and cofinite subsets form a Boolean Algebra

A set is *co-finite* if its 'co'mplement is finite.

- E.g., The universal set is cofinite since its complement is the emptyset, which is finite.
- E.g., The set of even numbers is neither finite nor cofinite, since its complement is the infinite set of odd numbers.

We know that all subsets form a Boolean Algebra, if we restrict ourselves to only those subsets that are finite or cofinite, then we only need to ensure that the set operators take 'good' sets to 'good' sets. Where,

- A set will, for ease of discussion, be qualified *good* if it is finite or cofinite.

Recall the grammar for set terms,

τ := $\varnothing$ | $\tau \cup \tau$ | $\tau \cap \tau$ | ~ τ | $\mathcal{U}$ | X where X is any good subset of the universal set $\mathcal{U}$

We only need to ensure that the constructions preserve the 'good' property:

- $\varnothing$: The empty set is good since it is finite.
- X $\cup$ Y: Assuming both X and Y are good, we must show the union is also a good set.
  - Case 1: Both X, Y are finite: Then their union is also finite, whence 'good'.

- Case 2: One of `X,Y` is not finite: Since they are good sets and non-finite, then one of them, say `X`, is necessarily cofinite and so its complement `~X` is finite. Then the complement of the union is `~(X ∪ Y) = ~X ∩ ~Y` is the intersection with a finite set `~X` and so the whole thing is necessarily finite. Whence, the the complement of the union is finite and so the union is a good set.
- Notice that the cases are exhaustive by the Law of the Excluded Middle: $p \vee \neg\, p \equiv \mathit{true}$.

- `X ∩ Y`: Assuming both `X` and `Y` are good, we must show the union is also a good set.
  - Case 1: One of them, say `X`, is finite: Then the intersection is necessarily finite and thus is 'good'.
  - Case 2: Neither are finite: Since they are good and not finite, they must both be cofinite and so their complements are finite sets. Thus, the complement of the intersection is the union of finite sets and so is a finite set. Hence, it is also a good set.
  - Notice that the cases are exhaustive by the Law of the Excluded Middle: $p \vee \neg\, p \equiv \mathit{true}$.

- `~ X`: Assume `X` is a good set.
  - Case 1: `X` is finite: Then the complement of `~ X` is `~ ~ X = X` is a finite set, whence `~ X` is a good set.
  - Case 2: `X` is cofinite: Then the complement of `X` is finite and so `~ X` is a good set.
  - Notice that the cases are exhaustive by the assumption of `X` being a good set.

- $\mathcal{U}$: The complement of the universal set is the empty set, which is finite, and so $\mathcal{U}$ is 'good'.

- `X`: If `X` is assumed to be a good set, then it remains a good set since we have not altered it in this construction.

10 When is my operation idempotent?

Suppose we have have a pair of operations $(\oplus,\otimes)$ where $\oplus$ distributes over $\otimes$ —i.e., $x \oplus (y \otimes z) = (x \oplus y) \otimes (x \oplus y)$— and a (one-sided) identity, say $\mathfrak{z}$, of $\oplus$ is a (one-sided) zero of $\otimes$ —i.e., $x \oplus \mathfrak{z} = x$ and $\mathfrak{z} = x \otimes \mathfrak{z}$— then we aim to show that $\otimes$ is necessarily idempotent.

Indeed,

$$
\begin{array}{ll}
 & x \otimes x \\
= & \quad \{\ \mathfrak{z} \text{ is identity of } \oplus\ \} \\
 & (x \oplus \mathfrak{z}) \otimes (x \oplus \mathfrak{z}) \\
= & \quad \{\ \oplus \text{ distributes over } \oplus\ \} \\
 & x \oplus (\mathfrak{z} \otimes \mathfrak{z}) \\
= & \quad \{\ \mathfrak{z} \text{ is the zero of } \otimes\ \} \\
 & x \oplus \mathfrak{z} \\
= & \quad \{\ \mathfrak{z} \text{ is identity of } \oplus\ \} \\
 & x
\end{array}
$$

Boolean Algebras readily provide examples of such triples: $(\sqcup, \sqcap, \bot)$ and $(\sqcap, \sqcup, \top)$.
Consequently, both meet and join are idempotent.

11 De Morgan's

The complement of x is the largest y that is disjoint from it, i.e., $y \sqcap x \sqsubseteq \bot \quad \equiv \quad y \sqsubseteq \bar{x}$.

- Challenge: Prove that, in the setting of Boolean Algebras, this is equivalent to saying $\bar{x} = y \quad \equiv \quad x \sqcap y = \bot \quad \wedge \quad x \sqcup y = \top$. In-fact this even holds in the setting of bounded distribute lattices.
- By duality, the complement of x is the smallest y that is exhaustive with it, i.e., $\top \sqsubseteq y \sqcup x \quad \equiv \quad \bar{x} \sqsubseteq y$.

Anyhow, we show $\overline{a \sqcup b} = \bar{a} \sqcap \bar{b}$ by showing, for any y, that $y \sqcap (a \sqcup b) \sqsubseteq \bot \quad \equiv \quad y \sqsubseteq \bar{a} \sqcap \bar{b}$.

$$
\begin{array}{ll}
& y \sqcap (a \sqcup b) \sqsubseteq \bot \\
= & \quad \{ \text{ Boolean algebras have meets and join distribute over one another } \} \\
& (y \sqcap a) \sqcup (y \sqcap b) \sqsubseteq \bot \\
= & \quad \{ \text{ Join characterisation } \} \\
& y \sqcap a \sqsubseteq \bot \quad \wedge \quad y \sqcap b \sqsubseteq \bot \\
= & \quad \{ \text{ Complement characterisation: Largest disjoint element } \} \\
& y \sqsubseteq \bar{a} \quad \wedge \quad y \sqsubseteq \bar{b} \\
= & \quad \{ \text{ Meet characterisation } \} \\
& y \sqsubseteq \bar{a} \sqcap \bar{b}
\end{array}
$$

Hence, $\overline{a \sqcup b} = \bar{a} \sqcap \bar{b}$.
By duality —i.e., replacing '$\sqcup, \sqsubseteq$' with '$\sqcap, \sqsupseteq$'—, we obtain the other variant of De Morgan's Law: $\overline{a \sqcap b} = \bar{a} \sqcup \bar{b}$.

12 The first n evens sum to $n \cdot (n + 1)$

We prove this result using the proof assistant CalcCheck:

```
Theorem "Even-number sum": (∑ i : ℕ | i ≤ n • 2 · i) = n · S n
Proof:
  By induction on 'n : ℕ':
    Base case:
        (∑ i : ℕ | i ≤ 0 • 2 · i)
      =⟨ "Zero is unique least element" ⟩
        (∑ i : ℕ | i = 0 • 2 · i)
      =⟨ "One-point rule" and Substitution ⟩
        2 · 0
      =⟨ "Zero of ·" and "Definition of ·" ⟩
        0 · S 0
    Induction step:
        (∑ i : ℕ | i ≤ S n • 2 · i)
      =⟨ "Split off term at top using ≤", Substitution ⟩
        (∑ i : ℕ | i ≤ n • 2 · i) + 2 · S n
      =⟨ Induction hypothesis ⟩
        n · S n + 2 · S n
      =⟨ "Distributivity of · over +" ⟩
        S n · (n + 2)
      =⟨ "Definition of +" and Evaluation - i.e., 2 = S S 0 ⟩
        S n · S S n
```

- The natural numbers are axiomatised in this system with zero `0 : ℕ` and successor function `S : ℕ → ℕ`.
- This system has been made available to `cas 701` students on Avenue.
- More on this system can be found at `http://calccheck.mcmaster.ca/`

13 Summing the first n squares

We prove this result using the proof assistant CalcCheck:

First some useful helpers,

```
Declaration: pow : ℕ → ℕ → ℕ
Axiom ‘‘Definition of ‘pow‘’’: pow x 0     = 1
Axiom ‘‘Definition of ‘pow‘’’: pow x (S n) = x · pow x n

Theorem ‘‘Multiplication twice is power 2’’: x · x = pow x 2
Proof: ?
```

Now the main result is proved by induction where in the induction step we massage both sides in an attempt to find a common middle expression that they both equal to. For the sake of readability, and elegance, we do so using sub-calculations.

```
Theorem ‘‘Sum the squares’’: 6 · (∑ i : ℕ  |  i ≤ n • pow i 2) = n · S n · (2 · n + 1)
Proof:
  By induction on ‘n : ℕ‘:
    Base case:
        6 · (∑ i : ℕ  |  i ≤ 0 • pow i 2)
      =⟨ ‘‘Zero is unique least element’’ ⟩
        6 · (∑ i : ℕ  |  i = 0 • pow i 2)
      =⟨ ‘‘One-point rule’’ and Substitution ⟩
        6 · pow 0 2
      =⟨ ‘‘Definition of ‘pow‘’’ and Evaluation - i.e., 2 = S S 0 ⟩
        6 · 0 · pow 0 1
      =⟨ ‘‘Zero of ·’’ ⟩
        0 · S 0 · (2 · 0 + 1)
    Induction step:
        6 · (∑ i : ℕ  |  i ≤ S n • pow i 2)
      =⟨ ‘‘Split off term at top using ≤’’, Substitution ⟩
        6 · ( (∑ i : ℕ  |  i ≤ n • pow i 2) + pow (S n) 2 )
      =⟨ ‘‘Distributivity of · over +’’ and Induction hypothesis ⟩
        n · S n · (2 · n + 1) + 6 · pow (S n) 2
      =⟨ ‘‘Multiplication twice is power 2’’ and ‘‘Distributivity of · over +’’ ⟩
        S n · ( n · (2 · n + 1) + 6 · S n )
      =⟨ Subproof for ‘n · (2 · n + 1) + 6 · S n = 2 · pow n 2 + 7 · n + 6‘:
            n · (2 · n + 1) + 6 · S n
          =⟨ ‘‘Distributivity of · over +’’ and ‘‘Definition of ·’’ ⟩
            n · 2 · n + n · 1 + 6 · n + 6
          =⟨ ‘‘Distributivity of · over +’’ and Evaluation ⟩
            n · 2 · n + 7 · n + 6
          =⟨ ‘‘Multiplication twice is power 2’’ ⟩
            2 · pow n 2 + 7 · n + 6
       ⟩
        S n · (2 · pow n 2 + 7 · n + 6)
      =⟨ Subproof for ‘S S n · (2 · S n + 1) = 2 · pow n 2 + 7 · n + 6‘:
            S S n · (2 · S n + 1)
          =⟨ ‘‘Definition of ·’’ ⟩
            (2 · S n + 1) + S n · (2 · S n + 1)
          =⟨ ‘‘Definition of ·’’ ⟩
            (2 · S n + 1) + (2 · S n + 1) + n · (2 · S n + 1)
          =⟨ ‘‘Distributivity of · over +’’ and ‘‘Definition of ·’’ ⟩
            2 + 2 · n + 1 + 2  + 2 · n + 1 + n · 2 + n · 2 · n + n · 1
```

```
      =⟨ ‘‘Distributivity of · over +’’ ⟩
        (2 + 2 + 2 + 1) · n + n · 2 · n + (2 + 1 + 2 + 1)
      =⟨ ‘‘Multiplication twice is power 2’’ and Evaluation ⟩
        2 · pow n 2 + 7 · n + 6
    ⟩
    S n · S S n · (2 · S n + 1)
```

- The natural numbers are axiomatised in this system with zero `0 : ℕ` and successor function `S : ℕ → ℕ`.
- This system has been made available to `cas 701` students on Avenue.
- More on this system can be found at `http://calccheck.mcmaster.ca/`

14 Bounding factorials by powers

We shall prove $n! < n^n$ for all naturals $n \geq 1$.

$$
\begin{array}{ll}
 & n! < n^n \\
= & \quad \{ \text{ Factorial of } n \text{ is the product of numbers } 1..n \\
 & \quad\ \ \text{Natural exponentiation is iterated product. } \} \\
 & (\Pi i : 1..n \bullet i) \quad < \quad (\Pi i : 1..n \bullet n) \\
\Leftarrow & \quad \{ \text{ Multiplication is strictly monotonic,} \\
 & \quad\ \ \text{so it suffices that the multiplicands observe the relation. } \} \\
 & \forall i : 1..n \bullet i < n \\
= & \quad \{ \text{ Type of } i \text{ trutifies the body of the forall } \} \\
 & \forall i : 1..n \bullet \mathsf{true} \\
= & \quad \{ \text{ Superflous quantification } \} \\
 & \mathsf{true}
\end{array}
$$

Challenge: Prove this inductively using the *CalcCheck* proof assistant.

15 Nibbles and bits with strings

Consider the grammar

```
S := ‘‘0’’
   | ‘‘0’’ S
   | S ‘‘0’’
   | ‘‘0’’ S ‘‘1’’
   | ‘‘1’’ S ‘‘0’’
```

Now define,

```
♯n t = the number of times bit n occurs in string t

♯0 ‘‘0’’             =   1
♯0 (‘‘0’’ s)         =   1 + ♯0 s
♯0 (s ‘‘0’’)         =   1 + ♯0 s
♯0 (‘‘0’’ s ‘‘1’’)   =   1 + ♯0 s
♯0 (‘‘1’’ s ‘‘0’’)   =   1 + ♯0 s

♯1 ‘‘0’’             =   0
♯1 (‘‘0’’ s)         =       ♯1 s
```

```
♯1 (s ‘‘0’’)        =      ♯1 s
♯1 (‘‘0’’ s ‘‘1’’)  =  1 + ♯1 s
♯1 (‘‘1’’ s ‘‘0’’)  =  1 + ♯1 s
```

We shall prove by structural induction that $\sharp_1 \mathtt{t} \leq \sharp_0 \mathtt{t}$, i.e., the number of 1's in a string is at-most the number of 0's it contains.

Case 1

$$
\begin{array}{ll}
 & \sharp_1 \text{“0”} \\
= & \quad \{ \text{ Definition } \} \\
 & 0 \\
\leq & \quad \{ \text{ Arithmetic } \} \\
 & 1 \\
= & \quad \{ \text{ Definition } \} \\
 & \sharp_0 \text{“0”}
\end{array}
$$

Case 2

$$
\begin{array}{ll}
 & \sharp_1(\text{“0”}s) \\
= & \quad \{ \text{ Definition } \} \\
 & \sharp_1 s \\
\leq & \quad \{ \text{ Induction hypothesis } \} \\
 & \sharp_0 s \\
\leq & \quad \{ \text{ Arithemtic } \} \\
 & 1 + \sharp_0 s \\
= & \quad \{ \text{ Definition } \} \\
 & \sharp_0(\text{“0”}s)
\end{array}
$$

Case 5

$$
\begin{array}{ll}
 & \sharp_1(\text{“1”}s\text{“0”}) \\
= & \quad \{ \text{ Definition } \} \\
 & 1 + \sharp_1 s \\
\leq & \quad \{ \text{ Induction hypothesis } \} \\
 & 1 + \sharp_0 s \\
= & \quad \{ \text{ Definition } \} \\
 & \sharp_0(\text{“1”}s\text{“0”})
\end{array}
$$

Case 3, for *s“0”*, is exceedingly similar to case 1; and case 4, for *“0”s“1”* is similar to case 5.

Challenge: Show that the grammar for S can be rewritten in Backus-Naur Form as

```
S := [[‘‘1’’] S] ‘‘0’’ [S [‘‘1’’]]
```