Prof. Bernd Finkbeiner, Ph.D.
Jana Hofmann, M.Sc.
Reactive Systems Group

Programmierung 1 (WS 2020/21)
Übungsblatt G

Lesen Sie im Buch Kapitel 7.7 - 9.5.

Hinweis: Über Aufgaben, die mit 🤔 markiert sind, müssen Sie eventuell etwas länger nachdenken. Falls Ihnen keine Lösung einfällt - kein Grund zur Sorge. Kommen Sie in die Office Hour, unsere Tutor:innen helfen gerne.

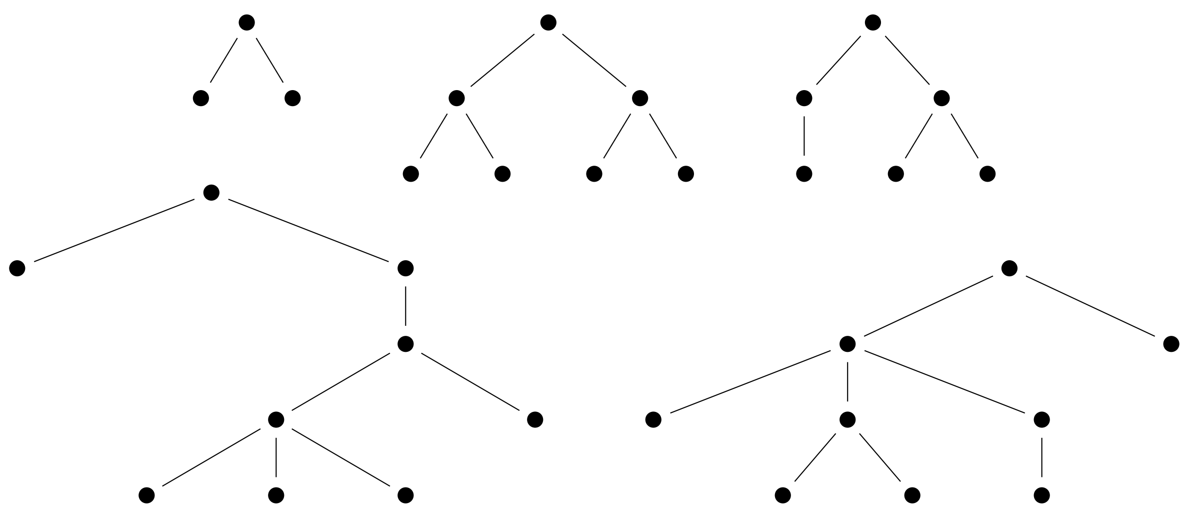
Diese Woche ist das Blatt etwas länger als gewöhnlich. Nutzen Sie die vorlesungsfreie Zeit indem Sie die entsprechenden Kapitel im Buch lesen und ausgiebig üben.

Bäume

Aufgabe G.1

(a) Geben Sie die Prä- und Postnummerierung der folgenden Bäume an.

(b) Geben Sie die Prä und Postlinearisierung der folgenden Bäume an.



Aufgabe G.2

Schreiben Sie eine Prozedur

(a) `pred : int list → int list`, die für eine gegebene Adresse den Vorgänger eines Knotens liefert. Wenn es sich bei dem Knoten um die Wurzel handelt, soll die Ausnahme `Subscript` geworfen werden.

(b) `succ : tree → int list → int → int list`, die für eine gegebene Adresse den k-ten Nachfolger eines Knotens liefert. Wenn der Knoten keinen k-ten Nachfolger hat, soll die Ausnahme `Subscript` geworfen werden.

Aufgabe G.3

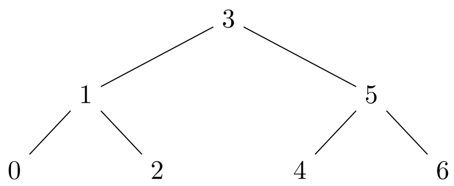
Schreiben Sie eine Prozedur `lfold : (α * β list → β) → α ltr → β`, die Faltung auf markierten Bäumen realisiert. Im Vergleich zu `fold` soll Ihre Prozedur `lfold` auch die Marken des Baumes mit einbeziehen (s. Typschema der Prozedur).

Aufgabe G.4

Bei der *Inprojektion* eines binären Baums erscheinen die Marken der inneren Knoten jeweils nach den Marken des linken und vor den Marken des rechten Unterbaums:

$$inpro\ (L\ (x, [t_1, t_2])) = inpro\ t_1\ @\ [x]\ @\ inprot_2$$

Beispielsweise ist $[0, 1, 2, 3, 4, 5, 6]$ die Inprojektion des Baums



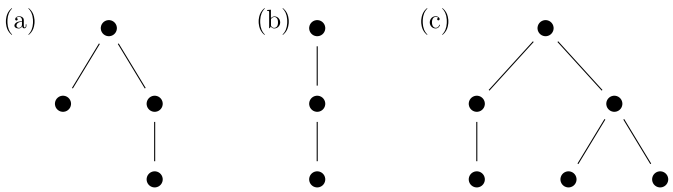
Schreiben Sie eine polymorphe Prozedur `inpro : α ltr → α list`, die die Inprojektion eines binären Baums liefert. Wenn die Prozedur auf einen Baum angewendet wird, der nicht binär ist, soll die Ausnahme `Subscript` geworfen werden.

Aufgabe G.5

Schreiben Sie eine Prozedur `sum : int ltr → int`, die die Summe aller Marken eines markierten `int`-Baumes berechnet. Es bietet sich an, dafür Ihre Prozedur `lfold` aus Aufgabe G.3 zu verwenden. Die Aufgabe kann aber auch ohne `lfold` gelöst werden.

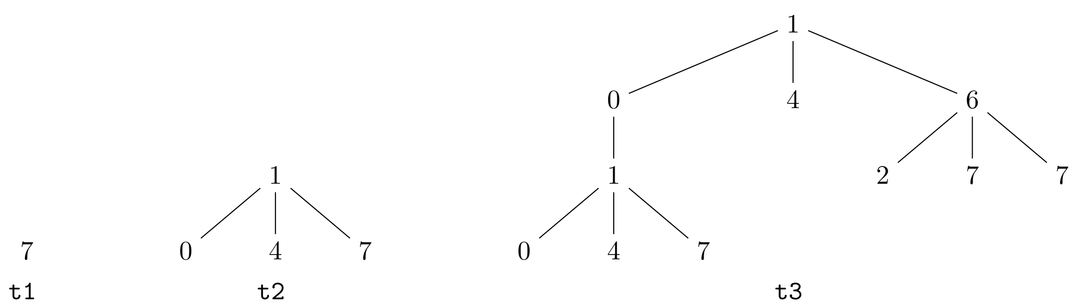
Aufgabe G.6

Geben Sie die finitären Mengen an, die durch folgende Bäume dargestellt werden.



Aufgabe G.7

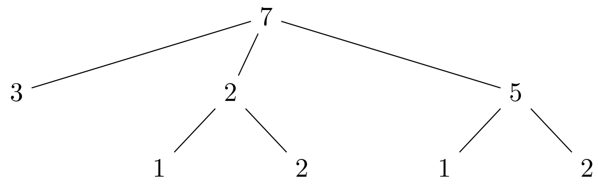
Deklarieren Sie eine Prozedur `compareLtr : (α * α → order) → α ltr * α ltr → order` die markierte Bäume vergleicht, indem eine Ordnung für die Markierungen mit der lexikalischen Baumordnung kombiniert wird. Beispielsweise sollen für die Bäume



die Aufrufe `compareLtr Int.compare (t1, t2)` und `compareLtr Int.compare (t1, t3)` den Wert `GREATER` liefern, und `compareLtr Int.compare (t2, t3)` soll `LESS` liefern.

Aufgabe G.8

Die n*-te Ebene* eines Baums ist die Liste der Marken der Knoten der Tiefe n, angeordnet entsprechend der Anordnung der Knoten. Als Beispiel betrachten wir den Baum



Die nullte Ebene ist $[7]$, die erste Ebene ist $[3, 2, 5]$, die zweite Ebene ist $[1, 2, 1, 2]$ und die dritte und alle nachfolgenden Ebenen sind $[]$. Schreiben Sie eine Prozedur `level : α ltr → int → α list`, die die n-te Ebene eines Baums liefert.

Mengen

Aufgabe G.9

Sei die Relation $R = \{(1, 2), (1, 3), (2, 4), (4, 1), (5, 2)\}$ gegeben.

(a) Zeichnen Sie die Graphdarstellung von R.

(b) Geben Sie die Mengen *Dom* R, *Ran* R und *Ver* R an.

(c) Ist R funktional? Injektiv? Total auf $\{1, 2, 4, 5\}$? Surjektiv auf *Ver* R?

(d) Welches Paar müssen Sie entfernen, damit R funktional und injektiv wird?

(e) Zeichnen Sie die Umkehrrelation R^{-1}.

(f) Zeichnen Sie die Komposition $R \circ R$.

Aufgabe G.10

Geben Sie die Elemente der folgenden Mengen an.

(a) $\mathbb{N}^* \cap \mathbb{B}^2$

(b) $\mathbb{B} \cap \mathbb{Z}$

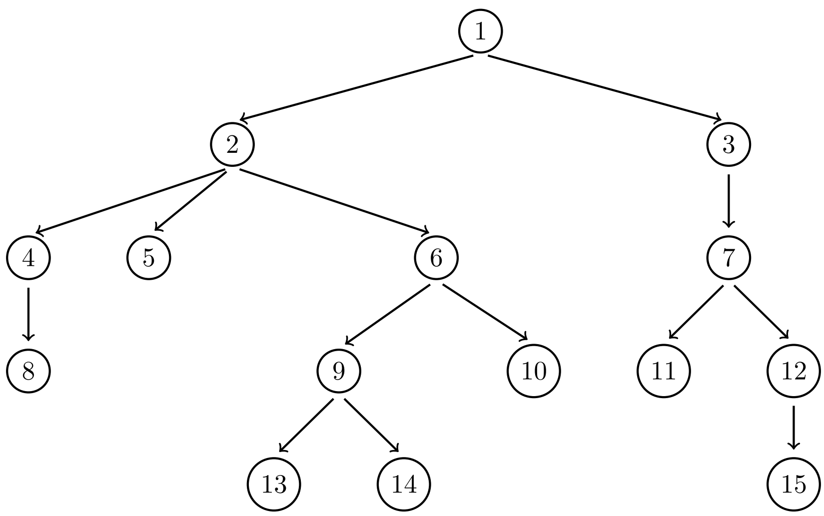
(c) $\mathbb{B} \cup (\mathbb{N} \cap [4, 6])$

(d) $(\mathbb{N} - [0, 4]) - \mathbb{B}$

(e) $\mathcal{P}(\mathbb{N} \cap [1, 3])$

Aufgabe G.11

Sei der folgende baumartige Graph gegeben:



(a) Geben Sie die Knotenmenge V und die Kantenmenge E des Graphen an.

(b) Geben Sie die Tiefe des Graphen an.

(c) Geben Sie die Wurzel des Graphen an.

(d) Geben Sie die terminalen Knoten des Graphen an.

(e) Geben Sie für den Knoten 14 einen Pfad an, der von der Wurzel zu diesem Knoten führt. Gibt es mehrere solcher Pfade?

(f) Zeichnen Sie alle Teilgraphen, die den Knoten 7 als Wurzel haben.

(g) Geben Sie einen Teilgraphen an, der nicht baumartig ist.

Aufgabe G.12
Warum ist die Inklusionsordnung $\subseteq$ für die Potenzmenge von $\{1, 2\}$ nicht linear?

Aufgabe G.13
Ist die Menge $\{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots\}$ wohlfundiert?

Aufgabe G.14
Geben Sie eine funktionale, injektive und nicht-terminierende Relation R an, so dass

$$\mathit{Ver}\ R = \{2n \mid n \in \mathbb{N}\}\,.$$

Beschreiben Sie R

(a) grafisch,

(b) mit Mengen-Notation,

(c) mit Lambda-Notation.

Aufgabe G.15
Beschreiben Sie mithilfe der Lambda-Notation:

(a) Eine unendliche Funktion, die nicht injektiv ist.

(b) Eine injektive Funktion $\mathbb{Z} \to \mathbb{N}$, die surjektiv auf $\mathbb{N}$ ist.

(c) Eine injektive Funktion $\mathbb{Z} \to \mathbb{N}$, die nicht surjektiv auf $\mathbb{N}$ ist.

(d) Die unendliche Folge $0, 2, 4, 6, \dots$ der geraden natürlichen Zahlen.

(e) Die unendliche Folge $4, 9, 16, 25, \dots$ der Quadrate der Zahlen ab 2.

(f) Die unendliche Folge $9, 25, 49, 81, \dots$ der Quadrate der ungeraden natürlichen Zahlen ab 3.

Eine Folge ist dabei definiert als eine Funktion aus $\mathbb{N} \to \mathbb{N}$, die für $n \in \mathbb{N}$ auf das $n$te Element der Folge abbildet.

Aufgabe G.16
Geben Sie jeweils eine Relation mit folgenden Eigenschaften an:

(a) Die Relation ist unendlich und terminiert für keinen ihrer Knoten.

(b) Die Relation ist unendlich, funktional und terminierend.

Aufgabe G.17
Gegeben die folgenden Datentypen zur Darstellung von Graphen:

```
type vertex = int
type edge = vertex * vertex
type graph = vertex list * edge list
```

Sei `g` vom Typ `graph`, wobei wir annehmen, dass seine Listen keine Duplikate enthalten.

Wir nennen `g` gültig, wenn `g` einen Graph repräsentiert, beispielsweise ist `([1, 2, 3], [(1,2)])` gültig, `([1], [(2,3)])` aber nicht.

(a) Schreiben Sie eine Prozedur `valid : graph` → `bool`, die prüft, ob ein Graph gültig ist.

(b) Schreiben Sie eine Prozedur `succ : vertex` → `graph` → `vertex list`, die für einen Knoten in Graph `g` alle seine Nachfolger in `g` liefert.

Aufgabe G.18
Geben Sie eine zweielementige Menge mit allen folgenden Eigenschaften an: 🤔

(a) Jede Konstituente der Menge ist ein Element der Menge.

(b) Jede nichtleere Konstituente der Menge ist eine einelementige Menge.

Können Sie auch eine drei- und eine vierelementige Menge mit diesen Eigenschaften angeben?

Mathematische Prozeduren

Aufgabe G.19
Bleiben die Wohlgeformtheitsbedingungen für die definierenden Gleichungen gültig, wenn man bei der Prozedur (Buch S. 180)

(a) *fac* den Ergebnisbereich zu $\mathbb{Z}$ verändert?

(b) *fac* den Ergebnisbereich zu $\mathbb{N}_+$ verändert?

(c) *fac* den Argumentbereich und den Ergebnisbereich zu $\mathbb{Z}$ verändert?

(d) *fac*$'$ den Argumentbereich zu $\mathbb{N}$ verändert?

(e) *euclid* den Ergebnisbereich zu $\mathbb{N}_+$ verändert?

(f) *gcd* den Argumentbereich zu $\mathbb{N} \times \mathbb{N}$ und den Ergebnisbereich zu $\mathbb{N}$ verändert?

Aufgabe G.20
Zu einer Prozedur $p : X \to Y$ kann man Prozeduren $X \to \mathbb{N}$ angeben, die für terminierende Argumente x von p die Größe und die Tiefe des Rekursionsbaums für x und p liefern.

(a) Schreiben Sie solche Prozeduren für *gcd*. Realisieren Sie die Prozeduren in Standard ML.

(b) Schreiben Sie eine solche Prozedur für *euclid*. Realisieren Sie die Prozeduren in Standard ML.

Aufgabe G.21
Sei eine Prozedur mit der folgenden Rekursionsfunktion r gegeben: 🤔

$$\lambda x \in \mathbb{Z}.\ \text{if } x < 0 \text{ then } \langle x - 5 \rangle \text{ else if } x < 4 \text{ then } \langle \rangle \text{ else } \langle x - 3, x - 2 \rangle$$

(a) Geben Sie den Argumentbereich der Prozedur an.

(b) Geben Sie den Rekursionsbaum für das Argument 8 an.

(c) Geben Sie eine nicht-rekursive Beschreibung der Terminierungsmenge T_n für $n \geq 0$ an. Dabei umfasst T_n alle terminierenden Argumente mit Rekursionstiefe höchstens n.

(d) Geben Sie den Definitionsbereich der Prozedur an.

kNobelpreis

Das kNobeltutorium zur letzten Aufgabe wird erst im neuen Jahr stattfinden.

Trotz der Pandemie ist Ihr Freund Dieter nun in Weihnachtsstimmung verfallen. Und wie! Seit Tagen ist er auf der Suche nach der perfekten Weihnachtsdekoration. Auf dem Dachboden seiner Großeltern ist er dabei auf eine verstaubte Kiste gestoßen, in der sich unendlich lange Lichterketten befinden.

Dieters Begeisterung über seinen Fund wandelt sich aber schnell in Frustration, als er bemerkt, dass viele der Lampen durch die lange Lagerung kaputt gegangen sind. Das alleine ist noch kein Problem, doch keine der Ketten scheint Dieters rätselhaften Vorstellungen zu entsprechen.

Am Morgen des 24. Dezembers klingelt Dieter Sie aus Ihrem verdienten Ferienschlaf: „Ich muss heute Plätzchen für meine Großeltern backen. Kannst du für mich die Stellung halten und Lichterketten testen?“ Vor Verwirrung und Müdigkeit kommen Sie gar nicht dazu, ihm zu widersprechen. „Sieh her, ich habe eine Maschine gebaut. Damit ist das Testen ganz einfach!“ Dieter zeigt stolz auf die mit Tannenzweigen und Sternen geschmückte Maschine und legt zur Demonstration eine Lichterkette in deren Trichter. Die Maschine dudelt ein paar Takte von *Jingle Bells*, die jedoch abrupt abbrechen. Dafür blinkt der größte Stern rot. „Das heißt, dass die Kette nicht passt. Sonst würde der Stern grün blinken. Ganz einfach, siehst du?“ Noch immer schlaftrunken blinzeln Sie Dieter an. „Toll, vielen Dank! Bring mir die Kette einfach vorbei, sobald du eine gefunden hast!“ Und schon ist Dieter verschwunden.

Wir modellieren die Lichterketten als terminierende Prozeduren `int → bool`, die für eine natürliche Zahl n angeben, ob die n-te Lampe der Kette noch leuchtet. Dieters Maschine stellen wir als eine Prozedur `(int → bool) → bool` dar, die für eine gegebene Lichterkette entscheidet, ob sie Dieters Bedürfnissen entspricht. Natürlich terminiert Dieters Maschine egal welche Lichterkette auch in diese hineingegeben wird; wie sollte Dieter sonst auch wissen, ob die Lichterkette seinen Vorstellungen entspricht falls die Maschine nicht terminiert?

Sie testen eine Kette aus der Kiste nach der anderen. Nachdem Sie mehrere Stunden vergeblich auf ein positives Ergebnis gehofft haben und die Zeiger der Uhr immer weiter in Richtung Heiligabend wandern, reicht es Ihnen. Sie beschließen, stattdessen selbst eine Kette zu bestimmen, die von der Maschine erkannt wird. Dazu bauen Sie eine Supermaschine, die für eine Maschine wie die, die Dieter Ihnen gegeben hat, eine geeignete Lichterkette konstruiert, falls eine solche existiert. Andernfalls kann eine beliebige Lichterkette gebaut werden, doch Ihre Supermaschine muss immer zum Ende kommen. Schließlich wartet schon Ihre Familie zum Weihnachtsessen auf Sie!

Aufgabe G.22

Schreiben Sie eine Prozedur `super : ((int → bool) → bool) → (int → bool)`, die Ihre Supermaschine implementiert. Insbesondere soll für eine Maschine `m` gelten: `m (super m) = true`, falls die Maschine irgendeine Lichterkette akzeptiert. Ihre Prozedur muss immer terminieren. Sie dürfen dafür annehmen, dass auch die Argumentprozedur auf allen Lichterketten als Argument terminiert.

Hinweis: Als Sie auf die Idee kommen Lichterketten in die Maschine zu geben, die Ausnahmen werfen, sprühen Funken von der Maschine und die Lichterkette scheint zu explodieren. Sie entfernen die Reste der Lichterkette rechtzeitig und glücklicherweise scheint die Maschine aber weiterhin zu funktionieren. Da Sie wissen Dieter würde es Ihnen nie verzeihen, falls Sie seine Maschine zerstören, entschließen Sie sich vollständig auf Seiteneffekte, insbesondere Ausnahmen und Speicher zu verzichten.