Natural induction
(a model-theoretic study)

SINGLE-SIDED DRAFT

Anders E.V. Lundstedt

Tue Jun 10 03:52:33 CEST 2025

Contents

5 Finitely non-standard models of Robinson arithmetic

§ 5.1 Introduction

¶ 5.1.1 To the best of my knowledge, there is no systematic study of finitely non-standard models of Robinson arithmetic—that is, of models of Robinson arithmetic with a non-empty finite set of non-standard numbers. This is a modest attempt at such a systematic study. My motivation for this study was to set the stage for its follow-up study (Ch. 6) of finitely non-standard models of weak extensions of Robinson arithmetic—a study that in turn was motivated by me expecting its applicability in Ch. 7. (My expectations of applicability were met: in Ch. 7 all counterexamples to inductiveness are finitely non-standard models, and when constructing these counterexamples I was helped by the results in this chapter and the next.)

¶ 5.1.2 I claim no novelty—nor do I claim that I present nothing novel. Perhaps any novelty mainly lies in the systematic exposition. Perhaps there is some value in that all methods used are simple—I think readers with, say, a proper undergraduate-level education in logic should have few problems following along.

¶ 5.1.3 While I have found no systematic study, the existing literature has examples of finitely non-standard models of Robinson arithmetic.

- In their classical textbook, Boolos and Jeffrey (1980) present a non-standard model of Robinson arithmetic as a hint to exercise 14.2. That non-standard model has two non-standard numbers and is thus finitely non-standard.

- I have seen a number of examples of models of Robinson arithmetic with one or two non-standard numbers. These models are thus finitely non-standard. Most likely Robinson himself presented a finitely non-standard model of Robinson arithmetic in his address at the 1950 International Congress of Mathematicians.[*] The abstract for that address includes the (presumably) original axiomatization of Robinson arithmetic, and the following parenthetical:

> (On the other hand, many simple formulas, such as $0+a = a$ and $a \leq a$, are not provable from the given axioms.) [Robinson (1950)]

One may easily show that $0+a = a$ is not provable from (Robinson's original axiomatization of) Robinson arithmetic by exhibiting a suitable countermodel with two non-standard numbers. (Readers might find proving thus to be a suitable warm-up for the material in this chapter.[†])

¶ 5.1.4 We recall Robinson arithmetic.

¶ 5.1.5 **Definitions 2.1.1 (restated)**

(a) The language $\mathcal{L}^{\mathrm{Q}}$ of Robinson arithmetic is the $\mathcal{L}^{\infty}$-reduct $\langle 0, S, +, \times \rangle$.

(b) The $\mathcal{L}^{\mathrm{Q}}$-theory Robinson arithmetic, notation '**Q**', is axiomatized by the respective universal closures of:

(Q1) $Sx \neq 0$

(Q2) $Sx = Sy \rightarrow x = y$

(Q3) $x = 0 \vee \exists y\, x = Sy$

(Q4) $x + 0 = x$

(Q5) $x + Sy = S(x + y)$

[*] No accompanying paper seem to have been published—at least it seems so according to the answers to a MathOverflow question (Brox, 2010) regarding exactly this. In any case, Robinson arithmetic is introduced and studied starting with section 3 of paper II in the monograph Undecidable Theories (Tarski, Mostwoski, Robinson, 1953, p. 51). Presumably what Robinson presented in his address was used in that paper.

[†] Another suitable warm-up exercise might be to prove that there is—up to isomorphism—exactly countably many distinct models of Robinson arithmetic with a single non-standard number c. (Hint: There is one such model—of Robinson's original axiomatization—for each possible value of $0 \times c$, which may be set to any standard number, or to the single non-standard number c.) When proof-reading, I found proving thus to be a suitable reminder of this chapter's ideas.

(Q6) $x \times 0 = 0$

(Q7) $x \times Sy = x \times y + x$.

¶ 5.1.6 For reasons given in ¶ 5.1.14, we also work with fragments of Robinson arithmetic.

¶ 5.1.7 **Definitions [fragments of Robinson arithmetic]**

(a) The language $\mathcal{L}^{\mathrm{p}}$ is the $\mathcal{L}^{\mathrm{Q}}$-reduct $\langle 0, S \rangle$.

(b) The language $\mathcal{L}^{+}$ is the $\mathcal{L}^{\mathrm{Q}}$-reduct $\langle 0, S, + \rangle$.

(c) The progression fragment (of Robinson arithmetic), notation '$\mathbf{Q}^{\mathrm{p}}$', is the $\mathcal{L}^{\mathrm{p}}$-theory axiomatized by (Q1)–(Q3).

(d) The addition fragment (of Robinson arithmetic), notation '$\mathbf{Q}^{+}$', is the $\mathcal{L}^{+}$-theory axiomatized by (Q1)–(Q5).

¶ 5.1.8 **Remark** My choice of the terminology 'progression fragment' is inspired by the observation made by Quine—made independently by others as well, I presume—that any progression will do as the set of natural numbers:

> The subtle point is that any progression will serve as a version of number so long and only so long as we stick to one and the same progression. Arithmetic is, in this sense, all there is to number: there is no saying absolutely what the numbers are; there is only arithmetic.
> [Quine, W.V (1968, p. 198)]

(Of course, the progression fragment of Robinson arithmetic admits of many models that are not the natural number progression—but then so does true first-order arithmetic. I think 'the progression fragment' makes for decent terminology—and after all, it is only terminology.)

¶ 5.1.9 **Definition [finitely non-standard models]** For $L = \mathcal{L}^{\mathrm{p}}$, $L = \mathcal{L}^{+}$ and $L = \mathcal{L}^{\mathrm{Q}}$, an L-model is finitely non-standard if and only if:

- its domain is $\mathbb{N} + A$ for some finite non-empty set A of non-standard numbers disjoint from $\mathbb{N}$; and

– restricting the domain to $\mathbb{N}$ is possible and this restriction is the standard L-model.

¶ 5.1.10 **Abbreviation** 'f.n.s.' abbreviates 'finitely non-standard'.

¶ 5.1.11 **Convention** When I use 'f.n.s. model'—without specifying a language—what I mean is an f.n.s. model of any of the three languages $\mathcal{L}^{\mathrm{p}}$, $\mathcal{L}^{+}$ and $\mathcal{L}^{\mathrm{Q}}$, to the respective extent a model of each language makes sense in the given context. This convention also applies, mutatis mutandis, in similar cases where no language is specified.

¶ 5.1.12 **Remarks**

(a) We could of course generalize '— is an f.n.s. model' by accounting for isomorphic models. There is no need to do this for present purposes.

(b) Note that we do not require an f.n.s. $\mathcal{L}^{\mathrm{Q}}$-model to be a model of $\mathbf{Q}$. Similarly, an f.n.s. $\mathcal{L}^{\mathrm{p}}$-model need not be a model of $\mathbf{Q}^{\mathrm{p}}$, and an f.n.s. $\mathcal{L}^{+}$-model need not be a model of $\mathbf{Q}^{+}$.

¶ 5.1.13 **Main results** The main results of this chapter are Facts 5.2.2, 5.4.6 and 5.5.14.* Together, these roughly say that for each f.n.s. $\mathcal{L}^{\mathrm{Q}}$-model of $\mathbf{Q}$:

– S is a permutation of the set of non-standard numbers. As is well-known, each permutation of a finite set has a unique "decomposition" into "cycles" on "disjoint orbits". In § 5.4, I define the preceding scare-quoted notions. I call the unique decomposition the (successor) cycle structure (of the model) and I call the cycles on disjoint orbits the (successor) cycles (of the model). (While this terminology may be non-standard, the definitions should not differ from how these notions are usually defined.)

– The restriction of $+$ to

$$\{\langle a, n\rangle \ :\ a \text{ non-standard, } n \text{ standard}\}$$

* Facts 5.4.24 and 5.5.36 are alternative formulations of Facts 5.4.6 and 5.5.14, respectively. These alternative formulations are a bit more informative and for some purposes more useful.

is "tame", in the following sense. This restriction is determined already by the cycle structure of the model—for each non-standard a and for each standard n we have

$$a + n = \text{the result of starting at } a \text{ and taking } n \text{ steps in its cycle.}$$

- The restriction of $+$ to

$$\{\langle \alpha, a \rangle \ : \ \alpha \text{ standard or non-standard, } a \text{ non-standard}\}$$

  is "wild". Contrary to the previous restriction, this one is not determined by the cycle structure of the model, but subject to some constraints—in particular, its range may only consist of non-standard numbers. Here we have quite some freedom when constructing an f.n.s. model of Q^+ that has more than a few non-standard numbers.

- Similar to the case of addition, the restriction of $\times$ to

$$\{\langle a, n \rangle \ : \ a \text{ non-standard, } n \text{ standard}\}$$

  is determined by S and $+$, whereas the restriction of $\times$ to

$$\{\langle \alpha, a \rangle \ : \ \alpha \text{ standard or non-standard, } a \text{ non-standard}\}$$

  is not uniquely determined (by S and $+$), but subject to some constraints.

¶ 5.1.14 I proceed in stages to establish the above characterization of f.n.s. $\mathcal{L}^{\mathsf{Q}}$-models of Q, with each stage building on the previous stage.

- In § 5.2 I deal with f.n.s. $\mathcal{L}^{\mathrm{p}}$-models.
- In § 5.3 I introduce some helpful conveniences and prove some helpful lemmas.
- In § 5.4 I deal with f.n.s. $\mathcal{L}^{+}$-models.
- In § 5.5 I deal with f.n.s. $\mathcal{L}^{\mathsf{Q}}$-models.

§ 5.2 Finitely non-standard models of the progression fragment of Robinson arithmetic

¶ 5.2.1 We recall the axiomatization of $\mathbf{Q}^{\mathrm{p}}$:

(Q1) $Sx \neq 0$

(Q2) $Sx = Sy \to x = y$

(Q3) $x = 0 \vee \exists y\, x = Sy$.

¶ 5.2.2 **Fact** An f.n.s. model is a model of $\mathbf{Q}^{\mathrm{p}}$ if and only if S restricted to the set of non-standard numbers is a permutation of the set of non-standard numbers.

¶ 5.2.3 **Proof** The if direction is trivial. For the only if direction, take any f.n.s. model that is a model of (Q1)–(Q3). By (Q1) and (Q2) and by the definition of '— is an f.n.s. model', none of our model's non-standard numbers has a standard successor. Thus since 0 is standard:

- By (Q3) each of our non-standard numbers has a non-standard predecessor.
- Thus (Q2) and (Q3) give that S restricted to the set of non-standard numbers is a permutation of the set of non-standard numbers.

¶ 5.2.4 **Fact** The $\mathcal{L}^{\mathrm{p}}$-reduct of each f.n.s. model of $\mathbf{Q}^{\mathrm{p}}$ is uniquely determined by the restriction of S to the model's set of non-standard numbers.

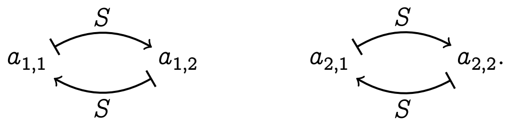
¶ 5.2.5 **Proof** For each f.n.s. model of $\mathbf{Q}^{\mathrm{p}}$, the definition of '— is an f.n.s. model' determines the interpretation of (the constant symbol) 0—namely, as (the number) 0—as well as the restriction of S to the standard numbers. Thus obviously the restriction of S to the set of non-standard numbers uniquely determines the $\mathcal{L}^{\mathrm{p}}$-reduct of each f.n.s. model of $\mathbf{Q}^{\mathrm{p}}$.

¶ 5.2.6 **Examples**

(a) Consider an f.n.s. $\mathcal{L}^{\mathrm{p}}$-model with a set A of non-standard numbers given by

$$A = A_1 + A_2 \quad A_1 = \{a_{1,1},\ a_{1,2}\} \quad A_2 = \{a_{2,1},\ a_{2,2}\}$$

(with the denotations of the '$a_{\_,\_}$' distinct from each other), and with $S{\downarrow}A$ given by



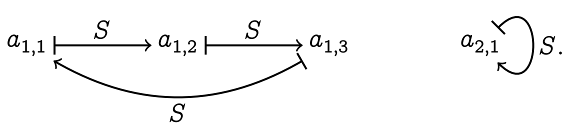
Clearly S is a permutation of A—that is, S is a permutation of the set of non-standard numbers. Thus, by Facts 5.2.2 and 5.2.4, the above defines a unique (up to isomorphism) f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$.

Note that each of A_1 and A_2 is closed under S and has no proper subset closed under S—these are the cycles of the model. The partition of A with A_1 and A_2 as its parts is the cycle structure of the model.

(b) Another f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$ with a set A of non-standard numbers is given by

$$A = A_1 + A_2 \quad A_1 = \{a_{1,1},\ a_{1,2},\ a_{1,3}\} \quad A_2 = \{a_{2,1}\},$$

with $S{\downarrow}A$ given by



¶ 5.2.7 Obviously, each f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$ is recursively representable (by Fact 5.2.2 and the definition of '— is an f.n.s. $\mathcal{L}^{\mathrm{p}}$-model'). Up to isomorphism, (recursive representations of) the f.n.s. $\mathcal{L}^{\mathrm{p}}$-models of $\mathbf{Q}^{\mathrm{p}}$ are also easy to recursively enumerate: for each positive integer n there is—up to isomorphism—as many f.n.s. $\mathcal{L}^{\mathrm{p}}$-models of $\mathbf{Q}^{\mathrm{p}}$ with n non-standard numbers as there are permutations of the set $\{1, \ldots, n\}$ (and by Facts 5.2.2 and 5.2.4, each such model is trivially recursively representable.)

§ 5.3 Some conveniences and some helpful lemmas

¶ 5.3.1 From here on, we work with an arbitrarily chosen $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$. We later (Assumption 5.4.2) expand it to an arbitrarily chosen $\mathcal{L}^{+}$-model of $\mathbf{Q}^{\mathrm{p}}$, which we in turn will (Assumption 5.5.2) expand to an arbitrary $\mathcal{L}^{\mathrm{Q}}$-model of $\mathbf{Q}^{+}$.

¶ 5.3.2 **Assumption** N_{p} is an arbitrarily chosen f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$.

¶ 5.3.3 I introduce some terminology, conventions, notations, definitions and results that will be useful when working with N_{p} and its upcoming expansions. In particular, Conventions 5.3.4 together with Remark 5.3.5(a) make good on my promise from ¶ 5.1.13 to make the scare-quoted notions precise in the well-known fact mentioned:

> Each permutation of a finite set has a unique "decomposition" into "cycles" on "disjoint orbits".

¶ 5.3.4 **Conventions**

(a) I denote the set of non-standard numbers of N_{p} by 'A'.

(b) By the definition of '— is an f.n.s. $\mathcal{L}^{\mathrm{p}}$-model', Fact 5.2.2, Assumption 5.3.2 and the well-known fact mentioned, there is a unique partition of the finite set of non-standard numbers of N_{p}—that is, of the set A—corresponding to the S-permutation of A: each part of this partition is a subset of A that is minimal with respect to closure under S (each part is closed under S but none of its proper subsets are). This partition is the (successor) cycle structure of N_{p}, and the parts of the partition are the (successor) cycles of N_{p}.

(c) By 'ν', I denote the number of cycles in the cycle structure.

(d) I use '$A_\_$', with indices denoting positive integers, to denote the cycles—that is, the ν cycles are:

$$A_1, \ldots, A_\nu.$$

(e) A cycle index is a natural number i such that $1 \leq i \leq \nu$.

(f) I use '$\mu[-]$' to denote the lengths of (that is, the set sizes of) the cycles—that is, for each cycle index i:

$$\mu[i] = \text{the length of cycle } A_i = \text{the size of } A_i.$$

(g) I use

$$`a_{-,1}\text{'}, \ `a_{-,2}\text{'}, \ `a_{-,3}\text{'}, \ \dots$$

to denote the (non-standard) numbers in a cycle—that is, for each cycle index i, the $\mu[i]$ non-standard numbers in A_i are:

$$a_{i,1}, \ \dots, \ a_{i,\mu[i]}.$$

¶ 5.3.5 **Remarks**

(a) Translating the (relevant parts of the) above into the terminology 'unique decomposition into cycles on disjoint orbits':

- A is closed under S and the structure $\langle A, S \downarrow A\rangle$ is a permutation of a finite set.
- For each cycle index i:
  - A_i is an orbit.
  - A_i is closed under S and the structure $\langle A_i, S \downarrow A_i\rangle$ is a cycle (on the orbit A_i).
- The unique decomposition of $\langle A, S \downarrow A\rangle$ (into cycles on disjoint orbits) is the union of the ν disjoint substructures

$$\langle A_1, S \downarrow A_1\rangle, \ \dots, \ \langle A_\nu, S \downarrow A_\nu\rangle.$$

(b) Note that Examples 5.2.6 used the notation, terminology and conventions just introduced. This was no coincidence: this lets us view these examples' $\mathcal{L}^{\mathrm{p}}$-models as concretizations of the arbitrarily chosen and thus indeterminately specified N_{p}.

(c) As N_{p} is assumed to be an arbitrarily chosen f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$, some of what was introduced by Conventions 5.3.4 may directly apply to other presentations of f.n.s. models of $\mathbf{Q}^{\mathrm{p}}$. For example, 'the cycle structure of' always applies. Other notions may require suitable reformulations.* For example 'cycle index' is not applicable to each presentation of an f.n.s. model.

* For an example of a reformulation, see Corollary 5.4.15 and its reformulation Example 5.4.17.

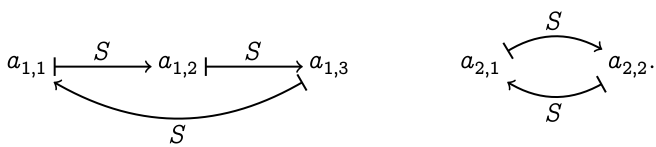
¶ 5.3.6 **Assumption** Without loss of generality, for each cycle index i:

$$\begin{aligned} Sa_{i,j} &= a_{i,j+1} \quad \text{if} \quad j < \mu[i] \\ Sa_{i,\mu[i]} &= a_{i,1}. \end{aligned}$$

¶ 5.3.7 **Example** Suppose we have a concretization of N_{p} given by:

$$A = A_1 + A_2 \quad A_1 = \{a_{1,1},\ a_{1,2},\ a_{1,3}\} \quad A_2 = \{a_{2,1},\ a_{2,2}\}.$$

Then, by Assumption 5.3.6, $S{\downarrow}A$ is given by:



¶ 5.3.8 **Definition** For each cycle index i:

$$\begin{aligned} a_i :& \quad \mathbb{Z} \to A_i \\ a_i(j) :=& \ a_{i,k} \quad \text{if and only if} \quad j \equiv k \mod \mu[i]. \end{aligned}$$

¶ 5.3.9 An example should illustrate the point of Definition 5.3.8.

¶ 5.3.10 **Example** Suppose $\mu[1] = 3$—that is:

$$A_1 = \{a_{1,1},\ a_{1,2},\ a_{1,3}\}.$$

We then have:

$$a_1(0) = a_{1,3}$$

$$\begin{array}{lcl} a_1(-1) = a_{1,2} & & a_1(1) = a_{1,1} \\ a_1(-2) = a_{1,1} & & a_1(2) = a_{1,2} \\ a_1(-3) = a_{1,3} & & a_1(3) = a_{1,3} \\ a_1(-4) = a_{1,2} & & a_1(4) = a_{1,1} \\ a_1(-5) = a_{1,1} & & a_1(5) = a_{1,2} \\ a_1(-6) = a_{1,3} & & a_1(6) = a_{1,3} \\ \vdots & & \vdots \end{array}$$

¶ 5.3.11 **Definition** The predecessor function on N_p, notation 'P', is defined by:

$$\begin{aligned} P:\quad & N_\mathrm{p} \to N_\mathrm{p} \\ P0 &:= 0 \\ Pn &:= n-1 && \text{if } n > 0 \text{ is standard} \\ Pa_{i,1} &:= a_{i,\mu[i]} \\ Pa_{i,j} &:= a_{i,j-1} && \text{if } j > 1 \end{aligned}$$

¶ 5.3.12 **Definitions**

(a) The iterated successor function on N_p, notation '$S^{-}-$', and the iterated predecessor function on N_p, notation '$P^{-}-$', are mutually defined by:

(1)
$$\begin{aligned} S^{-}-:\quad & \mathbb{Z} \times N_\mathrm{p} \to N_\mathrm{p} \\ S^{0}\alpha &:= \alpha \\ S^{n+1}\alpha &:= SS^{n}\alpha && \text{if } n \geq 0 \\ S^{n}\alpha &:= P^{n}\alpha && \text{if } n < 0 \end{aligned}$$

(2)
$$\begin{aligned} P^{-}-:\quad & \mathbb{Z} \times N_\mathrm{p} \to N_\mathrm{p} \\ P^{0}\alpha &:= \alpha \\ P^{n+1}\alpha &:= PP^{n}\alpha && \text{if } n \geq 0 \\ P^{n}\alpha &:= S^{n}\alpha && \text{if } n < 0 \end{aligned}$$

¶ 5.3.13 **Lemmas** For each standard n:

(a) – For each integer $m \geq -n$:

$$S^{m}n = n + m.$$

– For each integer $m < -n$:

$$S^{m}n = 0.$$

(b) – For each integer $m \leq n$:

$$P^{m}n = n - m.$$

– For each integer $m > n$:

$$P^{m}n = 0.$$

¶ 5.3.14 **Proofs** Intuitively follows from Definitions 5.3.12. I leave proofs as exercises for skeptic readers.

¶ 5.3.15 **Lemmas** For each non-standard $a_{i,j}$ and for each integer n:

(a) $S^n a_{i,j} = a_i(j+n)$

(b) $P^n a_{i,j} = a_i(j-n)$

(c) $a_{i,j} = S^n a_i(j-n)$

(d) $a_{i,j} = P^n a_i(j+n)$

(e) $a_{i,j} = S^{n\times\mu[i]} a_{i,j}$

(f) $a_{i,j} = P^{n\times\mu[i]} a_{i,j}$.

¶ 5.3.16 **Proofs** All follow from (Assumption 5.3.6 together with) the definitions of '$a\_$', 'S^-' and 'P^-' (Definition 5.3.8 and Definitions 5.3.12). A more detailed proof is in § 5.A, for readers who want it.

¶ 5.3.17 **Lemmas** For each standard n and for all integers k and m:

(a) We have

$$S^k S^m n = S^{k+m} n$$

if and only if:

- $m \geq -n$; or
- $m < -n$ and $k \leq 0$.

(b) We have

$$P^k P^m n = P^{k+m} n$$

if and only if:

- $m \leq n$; or
- $m > n$ and $k \geq 0$.

¶ 5.3.18 **Proofs** Intuitive and straightforward, but a bit tedious. I leave proofs as exercises for skeptic readers.

¶ 5.3.19 **Lemmas** For each non-standard a and for all integers k and m:

(a) $S^k S^m a = S^{k+m} a$

(b) $P^k P^m a = P^{k+m} a$.

¶ 5.3.20 **Proofs** See § 5.A.

¶ 5.3.21 I think most readers find Lemmas 5.3.13, 5.3.15, 5.3.17 and 5.3.19 all quite obvious and intuitive. For this reason, while I try to explicitly indicate each application of a previous result in my proofs, with these I make an exception—a reference to one of these lemmas would probably distract more than it would help.

¶ 5.3.22 **Lemma** For each non-standard $a_{i,j}$ and for each integer n, we have

$$S^n a_{i,j} = a_{i,j}$$

and

$$P^n a_{i,j} = a_{i,j}$$

if $\mu[i]$ divides n—otherwise we have neither.

¶ 5.3.23 As for Lemmas 5.3.13, 5.3.15, 5.3.17 and 5.3.19: Lemma 5.3.22 might be quite obvious to some readers, who thus may want to skip its proof. In any case, a proof is in § 5.A.

§ 5.4 Finitely non-standard models of the addition fragment of Robinson arithmetic

¶ 5.4.1 We recall the axiomatization of $\mathbf{Q}^+$:

(Q1) $Sx \neq 0$

(Q2) $Sx = Sy \rightarrow x = y$

(Q3) $x = 0 \vee \exists y\, x = Sy$

(Q4) $x + 0 = x$

(Q5) $x + Sy = S(x + y)$.

¶ 5.4.2 **Assumption** N_+ is an arbitrarily chosen $\mathcal{L}^+$-expansion of N_p.

¶ 5.4.3 For readers' convenience, we recall our previous assumption about N_p.

¶ 5.4.4 **Assumption 5.3.2 (restated)** N_p is an arbitrarily chosen f.n.s. $\mathcal{L}^\mathrm{p}$-model of $\mathbf{Q}^\mathrm{p}$.

¶ 5.4.5 **Remark** N_+ is thus an arbitrarily chosen f.n.s. $\mathcal{L}^+$-model of $\mathbf{Q}^\mathrm{p}$.

¶ 5.4.6 **Fact** N_+ is a model of $\mathbf{Q}^+$ if and only if:

(a) For each non-standard a and for each standard n:

$$a + n = S^n a.$$

(b) For each (standard or non-standard) α, for each cycle index i, and for each integer j:

$$\alpha + a_i(j) = S^{j-1}(\alpha + a_{i,1}).$$

¶ 5.4.7 **Remarks**

(a) Fact 5.4.6, modulo a suitable reformulation, applies to each f.n.s. model of $\mathbf{Q}^\mathrm{p}$. I formulated Fact 5.4.6 only for N_+ simply to have access to the convenient machinery introduced in § 5.3.

(b) I will continue in the style of Fact 5.4.6: results, definitions, et cetera will be formulated for N_p and its expansions, leaving more general reformulations implicit. (For pedagogical reasons Example 5.4.17 presents an explicit reformulation of Corollary 5.4.15.)

¶ 5.4.8 I split the proof of Fact 5.4.6 into two lemmas: Lemma 5.4.9 corresponds to Fact 5.4.6(a) and Lemma 5.4.11 corresponds to Fact 5.4.6(b).

¶ 5.4.9 **Lemma** (a) and (b) below are equivalent.

(a) (1) For each non-standard a:

$$N_+, [a/x] \models (Q4).$$

(2) For each non-standard a and for each standard n:

$$N_+, [a/x, n/y] \models (Q5).$$

(b) For each non-standard a and for each standard n:

$$a + n = S^n a.$$

¶ 5.4.10 **Proof** We have (a) if only and only if:

(0) $a + 0 = a$ for each non-standard a; and

(S) $a + n = S(a + (n-1))$ for each non-standard a and for each standard $n > 0$.

Some equivalence-preservering rewriting using (0) and (S) gives (b), thus completing the proof:

$$\begin{aligned} a + 0 &= a && \text{(by (0))} \\ &= S^0 a \end{aligned}$$

$$\begin{aligned} a + 1 &= S(a + 0) && \text{(by (S))} \\ &= SS^0 a && \text{(by previous)} \\ &= S^1 a \end{aligned}$$

$$\begin{aligned} a + 2 &= S(a + 1) && \text{(by (S))} \\ &= SS^1 a && \text{(by previous)} \\ &= S^2 a \\ a + 3 &= S(a + 2) && \text{(by (S))} \\ &= SS^2 a && \text{(by previous)} \\ &= S^3 a \end{aligned}$$

$$\vdots$$

¶ 5.4.11 **Lemma** (a) and (b) below are equivalent.

(a) For each (standard or non-standard) α and for each non-standard a:

$$N_+, [\alpha/x, a/y] \vDash \text{(Q5)}.$$

(b) For each (standard or non-standard) α, for each cycle index i, and for each integer j:

$$\alpha + a_i(j) = S^{j-1}(\alpha + a_{i,1}).$$

¶ 5.4.12 **Proof** We have (a) if and only if

$$\alpha + Sa = S(\alpha + a)$$

for each α and for each non-standard a. Thus (a) is equivalent to that for each α and each cycle index i:

$$\begin{aligned} \alpha + Sa_{i,1} &= S(\alpha + a_{i,1}) \\ &\vdots \\ \alpha + Sa_{i,\mu[i]} &= S(\alpha + a_{i,\mu[i]}). \end{aligned}$$

By the definition of '$a\_$', this system of equations is equivalent to:

$$\begin{aligned} &\vdots \\ \alpha + a_i(-2) &= S(\alpha + a_i(-3)) \\ \alpha + a_i(-1) &= S(\alpha + a_i(-2)) \\ \alpha + a_i(0) &= S(\alpha + a_i(-1)) \\ \alpha + a_i(1) &= S(\alpha + a_i(0)) \\ \alpha + a_i(2) &= S(\alpha + a_i(1)) \\ \alpha + a_i(3) &= S(\alpha + a_i(2)) \\ \alpha + a_i(4) &= S(\alpha + a_i(3)) \\ \alpha + a_i(5) &= S(\alpha + a_i(4)) \\ &\vdots \end{aligned}$$

Using the equivalence (under $\mathbf{Q}^{\mathrm{p}}$) between

$$\beta = S\gamma$$

and

$$\gamma = S^{-1}\beta,$$

we rewrite those equations that on their right hand side have a non-positive argument to a_i:

$$\vdots$$

(−3) $$\alpha + a_i(-3) = S^{-1}(\alpha + a_i(-2))$$

(−2) $$\alpha + a_i(-2) = S^{-1}(\alpha + a_i(-1))$$

(0) $$\alpha + a_i(-1) = S^{-1}(\alpha + a_i(0))$$

(1) $$\alpha + a_i(0) = S^{-1}(\alpha + a_i(1))$$

(2) $$\alpha + a_i(2) = S(\alpha + a_i(1))$$

(3) $$\alpha + a_i(3) = S(\alpha + a_i(2))$$

(4) $$\alpha + a_i(4) = S(\alpha + a_i(3))$$

(5) $$\alpha + a_i(5) = S(\alpha + a_i(4))$$

$$\vdots$$

We trivially have

$$\alpha + a_i(1) = S^{1-1}(\alpha + a_{i,1}),$$

which together with the following equivalence-preserving rewritings give (b), thus completing the proof.

– For (0), (−1), (−2), (−3), ... we have:

$$\begin{aligned}\alpha + a_i(0) &= S^{-1}(\alpha + a_i(1)) && \text{(by (0))}\\ &= S^{-1}(\alpha + a_{i,1})\\ &= S^{0-1}(\alpha + a_{i,1})\end{aligned}$$

$$\begin{aligned}\alpha + a_i(-1) &= S^{-1}(\alpha + a_i(0)) && \text{(by (−1))}\\ &= S^{-1}S^{0-1}(\alpha + a_{i,1}) && \text{(by previous)}\\ &= S^{-1-1}(\alpha + a_{i,1})\end{aligned}$$

$$\begin{aligned}\alpha + a_i(-2) &= S^{-1}(\alpha + a_i(-1)) && \text{(by (−2))}\\ &= S^{-1}S^{-1-1}(\alpha + a_{i,1}) && \text{(by previous)}\\ &= S^{-2-1}(\alpha + a_{i,1})\end{aligned}$$

$$\begin{aligned} \alpha + a_i(-3) &= S^{-1}(\alpha + a_i(-2)) && \text{(by } (-3)\text{)} \\ &= S^{-1}S^{-2-1}(\alpha + a_{i,1}) && \text{(by previous)} \\ &= S^{-3-1}(\alpha + a_{i,1}) \\ &\vdots \end{aligned}$$

– For (2), (3), (4), (5), ... we have:

$$\begin{aligned} \alpha + a_i(2) &= S(\alpha + a_i(1)) && \text{(by } (2)\text{)} \\ &= S(\alpha + a_{i,1}) \\ &= S^{2-1}(\alpha + a_{i,1}) \end{aligned}$$

$$\begin{aligned} \alpha + a_i(3) &= S(\alpha + a_i(2)) && \text{(by } (3)\text{)} \\ &= SS^{2-1}(\alpha + a_{i,1}) && \text{(by previous)} \\ &= S^{3-1}(\alpha + a_{i,1}) \end{aligned}$$

$$\begin{aligned} \alpha + a_i(4) &= S(\alpha + a_i(3)) && \text{(by } (4)\text{)} \\ &= SS^{3-1}(\alpha + a_{i,1}) && \text{(by previous)} \\ &= S^{4-1}(\alpha + a_{i,1}). \end{aligned}$$

$$\begin{aligned} \alpha + a_i(5) &= S(\alpha + a_i(4)) && \text{(by } (5)\text{)} \\ &= SS^{4-1}(\alpha + a_{i,1}) && \text{(by previous)} \\ &= S^{5-1}(\alpha + a_{i,1}) \\ &\vdots \end{aligned}$$

¶ 5.4.13 Fact 5.4.6 **(restated)** N_+ is a model of Q^+ if and only if:

(a) For each non-standard a and for each standard n:

$$a + n = S^n a.$$

(b) For each (standard or non-standard) α, for each cycle index i, and for each integer j:

$$\alpha + a_i(j) = S^{j-1}(\alpha + a_{i,1}).$$

¶ 5.4.14 **Proof** N_+ is a model of $\mathbf{Q}^+$ if and only if it is a model of (Q1)–(Q5). N_+ is a model of (Q1)–(Q3) since it is a model of $\mathbf{Q}^{\mathrm{p}}$. Thus N_+ is a model of $\mathbf{Q}^+$ if and only if it is a model of (Q4) and (Q5)—that is, if and only if:

($\mathrm{Q4_T}$) For each (standard or non-standard) number α:

$$N_+, [\alpha/x] \models \text{(Q4)}.$$

($\mathrm{Q5_T}$) For all (standard or non-standard) numbers α and β:

$$N_+, [\alpha/x, \beta/y] \models \text{(Q5)}.$$

The case α standard in ($\mathrm{Q4_T}$) and the case α and β standard in ($\mathrm{Q5_T}$) always hold by the definition of '— is an f.n.s. $\mathcal{L}^+$-model'. By Lemmas 5.4.9 and 5.4.11 the remaining cases hold if and only if we have (a) and (b).

¶ 5.4.15 **Corollary [of Fact 5.4.6]** Suppose $N_+ \models \mathbf{Q}^+$. Then N_+ is uniquely determined by its $\mathcal{L}^{\mathrm{p}}$-reduct together with the restriction of $+$ to

$$\{\langle \alpha, a_{i,1} \rangle \;:\; \alpha \text{ standard or non-standard, } i \text{ cycle index}\}.$$

¶ 5.4.16 **Proof** We need to show that this uniquely determines $+$. The definition of '— is an f.n.s. $\mathcal{L}^+$-model' determines $+$ on the standard numbers. Given the $\mathcal{L}^{\mathrm{p}}$-reduct and the given restriction of $+$, the remaining cases are determined by the equations in Fact 5.4.6.

¶ 5.4.17 **Example** Here follows a reformulation of Corollary 5.4.15 that applies to each f.n.s. model of $\mathbf{Q}^+$.

Consider any f.n.s. model M of $\mathbf{Q}^+$. For each cycle C of M, let $\gamma(C)$ be an arbitrarily chosen number from C. The $\mathcal{L}^+$-reduct of M is then uniquely determined by its $\mathcal{L}^{\mathrm{p}}$-reduct together with the restriction of $+$ to

$$\{\langle \alpha, \gamma(C) \rangle \;:\; \alpha \text{ standard or non-standard, } C \text{ cycle}\}.$$

¶ 5.4.18 **Fact** Suppose $N_+ \models \mathbf{Q}^+$. Then for each cycle index i and for each (standard or non-standard) α there is a cycle index k such that:

- $\alpha + a$ is in A_k for each a in A_i; and
- $\mu[k]$ divides $\mu[i]$.

¶ 5.4.19 For the purpose of elsewhere uses of it and elsewhere references to it, I prove the following lemma first.

¶ 5.4.20 **Lemma** Suppose $N_+ \models \mathbf{Q}^+$. Then for each (standard or non-standard) α and for all cycle indices i and k:

$$\alpha + a \text{ is in } A_k \text{ for some } a \text{ in } A_i$$

if and only if

$$\alpha + a \text{ is in } A_k \text{ for all } a \text{ in } A_i.$$

¶ 5.4.21 **Proof** Let $a_{i,j}$ and $a_{i,l}$ be any numbers in A_i. We have

$$\begin{aligned}
\alpha + a_{i,j} &= S^{j-1}(\alpha + a_{i,1}) && \text{(by Fact 5.4.6(b))}\\
&= S^{j-1}S^{0}(\alpha + a_{i,1})\\
&= S^{j-1}S^{l-1-(l-1)}(\alpha + a_{i,1})\\
&= S^{j-l}S^{l-1}(\alpha + a_{i,1})\\
&= S^{j-l}(\alpha + a_{i,l}) && \text{(ditto).}
\end{aligned}$$

Thus, since cycles are closed under (positive or negative iterations of) S, either both $\alpha + a_{i,j}$ and $\alpha + a_{i,l}$ are in A_k or none of them is.

¶ 5.4.22 **Proof [of Fact 5.4.18]** We have

(†) $$\alpha + a_{i,1} = S^{\mu[i]}(\alpha + a_{i,1})$$

by

$$\begin{aligned}
\alpha + a_{i,1} &= \alpha + Sa_{i,\mu[i]}\\
&= \alpha + Sa_i(\mu[i])\\
&= \alpha + a_i(\mu[i]+1)\\
&= S^{\mu[i]+1-1}(\alpha + a_{i,1}) && \text{(by Fact 5.4.6(b))}\\
&= S^{\mu[i]}(\alpha + a_{i,1}).
\end{aligned}$$

By (†), $\alpha + a_{i,1}$ must be non-standard (since $\mu[i] > 0$). Thus we have an $a_{k,j}$ such that:

(‡) $$\alpha + a_{i,1} = a_{k,j}.$$

- By (‡) and Lemma 5.4.20, $\alpha + a$ is in A_k for all a in A_i.
- Rewriting with (‡) in (†) we have

$$a_{k,j} = S^{\mu[i]} a_{k,j},$$

which by Lemma 5.3.22 gives that $\mu[k]$ divides $\mu[i]$.

¶ 5.4.23 Fact 5.4.18 is in a sense included in Fact 5.4.6: if we change '$a_i(j)$' to '$a_{i,j}$' in Fact 5.4.6(b), we need to add a divisibility condition, as in the following alternative formulation of Fact 5.4.6.

¶ 5.4.24 **Fact** N_+ is a model of $\mathbf{Q}^+$ if and only if:

(a) For each non-standard a and for each standard n:

$$a + n = S^n a.$$

(b) For each (standard or non-standard) α and for each non-standard $a_{i,j}$:

$$\alpha + a_{i,j} = S^{j-1}(\alpha + a_{i,1}).$$

(c) For each (standard or non-standard) α and for each cycle index i there is a cycle index k such that:

- $\alpha + a_{i,1}$ is in A_k.
- $\mu[k]$ divides $\mu[i]$.

¶ 5.4.25 **Remark** Note that it is only for $j = 1$ that Fact 5.4.24(c) requires that $\alpha + a_{i,j}$ is in a cycle of length dividing $\mu[i]$. Lemma 5.4.20—which may be proved using Fact 5.4.24(b) in the same way it was proved using Fact 5.4.6(b)—tells us why this works.

¶ 5.4.26 **Proof [of Fact 5.4.24]** The only if direction is immediate by Fact 5.4.6 and Fact 5.4.18. For the if direction, it clearly suffices to prove (the statement of) Fact 5.4.6(b)—that for each (standard or non-standard) α, for each cycle index i, and for each integer j:

$$\alpha + a_i(j) = S^{j-1}(\alpha + a_{i,1}).$$

Thus let α be any number, let i be any cycle index and let j be any integer. We have an $a_{i,k}$ and an integer n such that:

$$(\dagger) \qquad a_i(j) = a_{i,k}$$
$$(\ddagger) \qquad j = k + n \times \mu[i].$$

We then have

$$\begin{aligned} &\alpha + a_i(j) \\ &\quad = \alpha + a_{i,k} && \text{(by } (\dagger)\text{)} \\ &\quad = S^{k-1}(\alpha + a_{i,1}) && \text{(by Fact 5.4.24(b))} \\ &\quad = S^{k-1}S^{n\times\mu[i]}(\alpha + a_{i,1}) && \text{(by Lemma 5.3.22 and Fact 5.4.24(c))} \\ &\quad = S^{k+n\times\mu[i]-1}(\alpha + a_{i,1}) \\ &\quad = S^{j-1}(\alpha + a_{i,1}) && \text{(by } (\ddagger)\text{).} \end{aligned}$$

¶ 5.4.27 When constructing an f.n.s. $\mathcal{L}^+$-model of $\mathbf{Q}^+$, Fact 5.4.24 is more useful than Fact 5.4.6: from the former we may extract the following simple recipe for how to construct—up to isomorphism—any f.n.s. $\mathcal{L}^+$-model of $\mathbf{Q}^+$.

¶ 5.4.28 By Fact 5.4.24, up to isomorphism each f.n.s. $\mathcal{L}^+$-model of $\mathbf{Q}^+$ may be constructed by following the below instructions for how to turn our arbitrarily chosen f.n.s. $\mathcal{L}^+$-model N_+ of $\mathbf{Q}^\mathrm{p}$ into a concrete model of $\mathbf{Q}^+$.

(a) Choose a cycle structure for the $\mathcal{L}^\mathrm{p}$-reduct.

(b) For each (standard or non-standard) α and for each cycle index i: choose a non-standard a in a cycle of length dividing $\mu[i]$ and set

$$\alpha + a_{i,1} \coloneqq a.$$

(c) Refer to the equations in Fact 5.4.24 for how to define those remaining additions that involve non-standard numbers. (Those additions only involving standard numbers are of course as expected—and given by the definition of '— is an f.n.s. $\mathcal{L}^+$-model'.)

¶ 5.4.29 **Example** We follow the recipe in ¶ 5.4.28 to expand our $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$ from Example 5.2.6(a) to an $\mathcal{L}^{+}$-model of $\mathbf{Q}^{+}$.* Remark 5.3.5(b), it was no coincidence that we defined that model using some of the notation and conventions later introduced for N_{p}—thus letting us view it as a concretization of N_{p}. We recall the model:

$$\begin{aligned} A &= A_1 + A_2 \\ A_1 &= \{a_{1,1},\ a_{1,2}\} \\ A_2 &= \{a_{2,1},\ a_{2,2}\} \\ Sa_{1,1} &= a_{1,2} \\ Sa_{1,2} &= a_{1,1} \\ Sa_{2,1} &= a_{2,2} \\ Sa_{2,2} &= a_{2,1}. \end{aligned}$$

We follow the recipe.

(a) The first instruction of the recipe—"choose a cycle structure for the $\mathcal{L}^{\mathrm{p}}$-reduct"—is already taken care of (by Example 5.2.6(a)).

(b) For the second instruction, for each (non-standard or standard) α we should choose non-standard a and b and set

$$\begin{aligned} \alpha + a_{1,1} &\coloneqq a \\ \alpha + a_{2,1} &\coloneqq b, \end{aligned}$$

while ensuring that both a and b satisfy their respective divisibility requirements—but since both cycles are of equal length, the divisibility requirements will automatically be satisfied no matter our choices.

– For $\alpha = n$ standard, and for $i = 1$ and $i = 2$, we choose:

$$n + a_{i,1} \coloneqq S^n a_{i,1} \quad (= a_{i,1} \text{ if } n \text{ even, } a_{i,2} \text{ if } n \text{ odd}).$$

* § 5.C provides a Coq formalization which verifies that the thus obtained concretization of N_{+} indeed is a model of $\mathbf{Q}^{+}$.

– For $\alpha = a_{\_,\_}$ non-standard we choose:

$$\begin{aligned}
a_{1,1} + a_{1,1} &\coloneqq a_{1,1}\\
a_{1,2} + a_{1,1} &\coloneqq a_{1,2}\\
a_{2,1} + a_{1,1} &\coloneqq a_{2,1}\\
a_{2,2} + a_{1,1} &\coloneqq a_{2,2}\\
a_{1,1} + a_{2,1} &\coloneqq a_{2,2}\\
a_{1,2} + a_{2,1} &\coloneqq a_{2,2}\\
a_{2,1} + a_{2,1} &\coloneqq a_{2,1}\\
a_{2,2} + a_{2,1} &\coloneqq a_{2,1}.
\end{aligned}$$

(c) For the third instruction, we should refer to the equations in Fact 5.4.24 for how to define those remaining additions that involve non-standard numbers. To make our definition completely explicit, while not boring readers too much, I defer that to § 5.B.

¶ 5.4.30 **Fact** Each f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$ can be expanded to an f.n.s. $\mathcal{L}^{+}$-model of $\mathbf{Q}^{+}$.

¶ 5.4.31 **Proof** Each f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$ is isomorphic to one constructed following the recipe in ¶ 5.4.28. Thus consider a thus constructed f.n.s. $\mathcal{L}^{\mathrm{p}}$-model of $\mathbf{Q}^{\mathrm{p}}$. No matter the cycle structure chosen in ¶ 5.4.28(a), one may carry out ¶ 5.4.28(b): for each standard or non-standard α and for each cycle index i one may—to satisfy the divisibility requirement—simply choose an a in A_i. (Given that ¶¶ 5.4.28(a) and 5.4.28(b) have been carried out, ¶ 5.4.28(c) may always be carried out.)

¶ 5.4.32 **Fact** There are uncountably many non-isomorphic f.n.s. models of $\mathbf{Q}^{+}$.

¶ 5.4.33 **Proof** Consider concretizing N_{+} into a model of $\mathbf{Q}^{+}$ by following the recipe in ¶ 5.4.28. To carry out ¶ 5.4.28(a) we choose:

(†) $$A \coloneqq A_1 \coloneqq \{a_{1,1},\ a_{1,2}\}.$$

For purposes of this proof, it does not matter how we carry out ¶ 5.4.28(b) for the non-standard numbers—let us choose:

(‡) $$a_{1,1} + a_{1,1} \coloneqq a_{1,1} \eqqcolon a_{1,2} + a_{1,1}.$$

Next note the following freedom we have in completing ¶ 5.4.28(b): for each natural number n we may choose either $n + a_{1,1} := a_{1,1}$ or $n + a_{1,1} := a_{1,2}$, and no matter our choices the last step of the recipe (¶ 5.4.28(c)) may be carried out, and furthermore, for each set of such choices, it may be carried out in exactly one way. Thus there is a bijection between

$$\mathbb{N} \to \{a_{1,1},\ a_{1,2}\}$$

and the set of conretizations of N_+ that models Q^+ and that satisfy (†) and (‡). Since distinct such concretizations are non-isomorphic, we thus have uncountably many non-isomorphic f.n.s. models of Q^+.

¶ 5.4.34 **Open problem?** Is there a recursively enumerable set R of recursive presentations of f.n.s. models of Q^+ such that, up to isomorphism, each recursive f.n.s. model of Q^+ has a representation in R?

§ 5.5 Finitely non-standard models of Robinson arithmetic

¶ 5.5.1 We recall the axiomatization of Q:

(Q1) $Sx \neq 0$

(Q2) $Sx = Sy \to x = y$

(Q3) $x = 0 \lor \exists y\, x = Sy$

(Q4) $x + 0 = x$

(Q5) $x + Sy = S(x + y)$

(Q6) $x \times 0 = 0$

(Q7) $x \times Sy = x \times y + x$.

¶ 5.5.2 **Assumption** N is an arbitrarily chosen $\mathcal{L}^{\mathsf{Q}}$-expansion of N_+.

¶ 5.5.3 From here on we assume that N_+ is a model of Q^+.

¶ 5.5.4 **Assumption** $N_+ \vDash \mathsf{Q}^+$.

¶ 5.5.5 For readers' convenience, we recall our previous assumptions about N_+.

¶ 5.5.6 **Assumption 5.3.2 (restated)** N_p is an arbitrarily chosen f.n.s. $\mathcal{L}^\mathrm{p}$-model of $\mathbf{Q}^\mathrm{p}$.

¶ 5.5.7 **Assumption 5.4.2 (restated)** N_+ is an arbitrarily chosen $\mathcal{L}^+$-expansion of N_p.

¶ 5.5.8 **Remark** N is thus an arbitrarily chosen f.n.s. $\mathcal{L}^\mathrm{Q}$-model of $\mathbf{Q}^+$.

¶ 5.5.9 The notation provided by the following definition will be convenient.

¶ 5.5.10 **Definition** For each language L expanding $\mathcal{L}^+$, addition to the right, notation '$\oplus_{\_,\_}$', is defined for each number β in each L-model M:

$$\oplus_{\beta,M}: \quad M \to M$$
$$\oplus_{\beta,M}(\alpha) := \alpha + \beta.$$

¶ 5.5.11 **Notations**

- When possible, I allow myself to omit the second subscript in '$\oplus_{\_,\_}$'.
- When possible, I allow myself to omit the parentheses in '$\oplus_{\_,\_}(-)$'.

¶ 5.5.12 The purpose of Definition 5.5.10 is to provide a convenient notation for left-associative sums.

¶ 5.5.13 **Example** For all α and β in any $\mathcal{L}^\mathrm{Q}$-model M:

$$\begin{aligned}
\oplus_\beta^0\alpha &= \alpha \\
\oplus_\beta^1\alpha &= \alpha + \beta \\
\oplus_\beta^2\alpha &= (\alpha + \beta) + \beta \\
\oplus_\beta^3\alpha &= ((\alpha + \beta) + \beta) + \beta \\
&\vdots
\end{aligned}$$

¶ 5.5.14 **Fact** N is a model of **Q** if and only if:

(a) For each non-standard a and for each standard n:

$$a \times n = \oplus_a^n 0.$$

(b) For each standard n, for each cycle index i, and for each integer j:

$$n \times a_i(j) = S^{n \times (j-1)}(n \times a_{i,1}).$$

(c) For each non-standard a, for each cycle index i, and for each positive integer j:

$$a \times a_i(j) = \oplus_a^{j-1}(a \times a_{i,1}).$$

¶ 5.5.15 I split the proof of Fact 5.5.14 into three lemmas: Lemma 5.5.16 corresponds to Fact 5.5.14(a); Lemma 5.5.18 corresponds to Fact 5.5.14(b); Lemma 5.5.20 corresponds to Fact 5.5.14(c).

¶ 5.5.16 **Lemma** (a) and (b) below are equivalent.

(a) (1) For each non-standard a:

$$N, [a/x] \models (\text{Q6}).$$

(2) For each non-standard a and for each standard n:

$$N, [a/x, n/y] \models (\text{Q7}).$$

(b) For each non-standard a and for each standard n:

$$a \times n = \oplus_a^n 0.$$

¶ 5.5.17 **Proof** We have (a)(1) and (a)(2) if only and only if:

(0) $a \times 0 = 0$ for each non-standard a; and

(S) $a \times n = a \times (n-1) + a$ for each non-standard a and for each standard $n > 0$.

Some equivalence-preserving rewriting using (0) and (S) gives (b), thus completing the proof:

$$
\begin{aligned}
a \times 0 &= 0 && \text{(by (0))} \\
&= \oplus_a^0 0
\end{aligned}
$$

$$
\begin{aligned}
a \times 1 &= 0 + a \times 0 && \text{(by (S))} \\
&= \oplus_a^0 0 + 0 && \text{(by previous)} \\
&= \oplus_a^1 0
\end{aligned}
$$

$$
\begin{aligned}
a \times 2 &= 0 + a \times 1 && \text{(by (S))} \\
&= \oplus_a^1 0 + 0 && \text{(by previous)} \\
&= \oplus_a^2 0
\end{aligned}
$$

$$
\begin{aligned}
a \times 3 &= 0 + a \times 2 && \text{(by (S))} \\
&= \oplus_a^2 0 + 0 && \text{(by previous)} \\
&= \oplus_a^3 0
\end{aligned}
$$

$$\vdots$$

¶ 5.5.18 **Lemma** (a) and (b) below are equivalent.

(a) For each standard n and for each non-standard a:

$$N, [n/x, a/y] \models (\text{Q7}).$$

(b) For each standard n, for each cycle index i, and for each integer j:

$$n \times a_i(j) = S^{n \times (j-1)}(n \times a_{i,1}).$$

¶ 5.5.19 **Proof** We have (a) if and only if

$$n \times Sa = n \times a + n$$

for each standard n and for each non-standard a—that is, if and only if for each standard n and for each cycle index i:

$$\begin{aligned} n \times Sa_{i,1} &= n \times a_{i,1} + n \\ &\vdots \\ n \times Sa_{i,\mu[i]} &= n \times a_{i,\mu[i]} + n. \end{aligned}$$

This system of equations is equivalent to:

$$\begin{aligned} &\vdots \\ n \times a_i(-2) &= n \times a_i(-3) + n \\ n \times a_i(-1) &= n \times a_i(-2) + n \\ n \times a_i(0) &= n \times a_i(-1) + n \\ n \times a_i(1) &= n \times a_i(0) + n \\ n \times a_i(2) &= n \times a_i(1) + n \\ n \times a_i(3) &= n \times a_i(2) + n \\ n \times a_i(4) &= n \times a_i(3) + n \\ n \times a_i(5) &= n \times a_i(4) + n \\ &\vdots \end{aligned}$$

By Fact 5.4.6(a) the above is equivalent to:

$$\begin{aligned} &\vdots \\ n \times a_i(-2) &= S^n(n \times a_i(-3)) \\ n \times a_i(-1) &= S^n(n \times a_i(-2)) \\ n \times a_i(0) &= S^n(n \times a_i(-1)) \\ n \times a_i(1) &= S^n(n \times a_i(0)) \\ n \times a_i(2) &= S^n(n \times a_i(1)) \\ n \times a_i(3) &= S^n(n \times a_i(2)) \\ n \times a_i(4) &= S^n(n \times a_i(3)) \\ n \times a_i(5) &= S^n(n \times a_i(4)) \\ &\vdots \end{aligned}$$

Using the equivalence (under $\mathbf{Q}^{\mathrm{p}}$) between

$$\alpha = S^n \beta$$

and

$$\beta = S^{-n}\alpha,$$

we rewrite those equations that on their right hand side have a non-positive argument to a_i:

$$\vdots$$

$$
\begin{array}{ll}
(-3) & n \times a_i(-3) = S^{-n}(n \times a_i(-2)) \\
(-2) & n \times a_i(-2) = S^{-n}(n \times a_i(-1)) \\
(-1) & n \times a_i(-1) = S^{-n}(n \times a_i(0)) \\
(0) & n \times a_i(0) = S^{-n}(n \times a_i(1)) \\
(2) & n \times a_i(2) = S^{n}(n \times a_i(1)) \\
(3) & n \times a_i(3) = S^{n}(n \times a_i(2)) \\
(4) & n \times a_i(4) = S^{n}(n \times a_i(3)) \\
(5) & n \times a_i(5) = S^{n}(n \times a_i(4))
\end{array}
$$

$$\vdots$$

We trivially have

$$n \times a_i(1) = S^{n\times(1-1)}(n \times a_{i,1}),$$

which together with the following equivalence-preserving rewritings give (b), thus completing the proof:

- For (0), (−1), (−2), (−3), ... we have:

$$
\begin{aligned}
n \times a_i(0) &= S^{-n}(n \times a_i(1)) && \text{(by (0))} \\
&= S^{-n}(n \times a_{i,1}) \\
&= S^{n\times(0-1)}(n \times a_{i,1})
\end{aligned}
$$

$$
\begin{aligned}
n \times a_i(-1) &= S^{-n}(n \times a_i(0)) && \text{(by (−1))} \\
&= S^{-n}S^{n\times(0-1)}(n \times a_{i,1}) && \text{(by previous)} \\
&= S^{n\times(-1-1)}(n \times a_{i,1})
\end{aligned}
$$

$$
\begin{aligned}
n \times a_i(-2) &= S^{-n}(n \times a_i(-1)) && \text{(by (−2))} \\
&= S^{-n}S^{n\times(-1-1)}(n \times a_{i,1}) && \text{(by previous)}
\end{aligned}
$$

$$= S^{n\times(-2-1)}(n \times a_{i,1})$$

$$\begin{aligned}
n \times a_i(-3) &= S^{-n}(n \times a_i(-2)) && \text{(by (−3))}\\
&= S^{-n}S^{n\times(-2-1)}(n \times a_{i,1}) && \text{(by previous)}\\
&= S^{n\times(-3-1)}(n \times a_{i,1})
\end{aligned}$$

$$\vdots$$

– For (2), (3), (4), (5) ... we have:

$$\begin{aligned}
n \times a_i(2) &= S^{n}(n \times a_i(1)) && \text{(by (2))}\\
&= S^{n}(n \times a_{i,1})\\
&= S^{n\times(2-1)}(n \times a_{i,1})
\end{aligned}$$

$$\begin{aligned}
n \times a_i(3) &= S^{n}(n \times a_i(2)) && \text{(by (3))}\\
&= S^{n}S^{n\times(2-1)}(n \times a_{i,1}) && \text{(by previous)}\\
&= S^{n\times(3-1)}(n \times a_{i,1})
\end{aligned}$$

$$\begin{aligned}
n \times a_i(4) &= S^{n}(n \times a_i(3)) && \text{(by (4))}\\
&= S^{n}S^{n\times(3-1)}(n \times a_{i,1}) && \text{(by previous)}\\
&= S^{n\times(4-1)}(n \times a_{i,1})
\end{aligned}$$

$$\begin{aligned}
n \times a_i(5) &= S^{n}(n \times a_i(4)) && \text{(by (5))}\\
&= S^{n}S^{n\times(4-1)}(n \times a_{i,1}) && \text{(by previous)}\\
&= S^{n\times(5-1)}(n \times a_{i,1})
\end{aligned}$$

$$\vdots$$

¶ 5.5.20 **Lemma** (a) and (b) below are equivalent.

(a) For all non-standard a and b:

$$N, [a/x, b/y] \models \text{(Q7)}.$$

(b) For each non-standard a, for each cycle index i, and for each positive integer j:

$$a \times a_i(j) = \oplus_a^{j-1}(a \times a_{i,1}).$$

¶ 5.5.21 **Proof** We have (a) if and only if

$$a \times Sb = a \times b + a$$

for all non-standard a and b—that is, if and only if for each non-standard a and for each cycle index i:

$$\begin{gathered} a \times Sa_{i,1} = a \times a_{i,1} + a \\ \vdots \\ a \times Sa_{i,\mu[i]} = a \times a_{i,\mu[i]} + a. \end{gathered}$$

This system of equations is equivalent to:

$$a \times a_i(2) = a \times a_i(1) + a \tag{2}$$
$$a \times a_i(3) = a \times a_i(2) + a \tag{3}$$
$$a \times a_i(4) = a \times a_i(3) + a \tag{4}$$
$$a \times a_i(5) = a \times a_i(4) + a \tag{5}$$
$$\vdots$$

We trivially have

$$a \times a_i(1) = \oplus_a^{1-1}(a \times a_{i,1}),$$

which together with the following equivalence-preserving rewritings give (b), thus completing the proof.

$$\begin{aligned} a \times a_i(2) &= a \times a_i(1) + a && \text{(by (2))} \\ &= a \times a_{i,1} + a \\ &= \oplus_a^1(a \times a_{i,1}) \\ &= \oplus_a^{2-1}(a \times a_{i,1}) \end{aligned}$$

$$\begin{aligned} a \times a_i(3) &= a \times a_i(2) + a && \text{(by (3))} \\ &= \oplus_a^{2-1}(a \times a_{i,1}) + a && \text{(by previous)} \\ &= \oplus_a^{3-1}(a \times a_{i,1}) \end{aligned}$$

$$\begin{aligned} a \times a_i(4) &= a \times a_i(3) + a && \text{(by (4))} \\ &= \oplus_a^{3-1}(a \times a_{i,1}) + a && \text{(by previous)} \\ &= \oplus_a^{4-1}(a \times a_{i,1}) \end{aligned}$$

$$\begin{aligned} a \times a_i(5) &= a \times a_i(4) + a && \text{(by (5))} \\ &= \oplus_a^{4-1}(a \times a_{i,1}) + a && \text{(by previous)} \\ &= \oplus_a^{5-1}(a \times a_{i,1}) \end{aligned}$$

$$\vdots$$

¶ 5.5.22 Fact 5.5.14 **(restated)** N is a model of **Q** if and only if:

(a) For each non-standard a and for each standard n:

$$a \times n = \oplus_a^n 0.$$

(b) For each standard n, for each cycle index i, and for each integer j:

$$n \times a_i(j) = S^{n \times (j-1)}(n \times a_{i,1}).$$

(c) For each non-standard a, for each cycle index i, and for each positive integer j:

$$a \times a_i(j) = \oplus_a^{j-1}(a \times a_{i,1}).$$

¶ 5.5.23 **Proof** N is a model of **Q** if and only if it is a model of (Q1)–(Q7). N is a model of (Q1)–(Q5) since it is a model of $\mathbf{Q}^+$. Thus N is a model of **Q** if and only if it is a model of (Q6) and (Q7)—that is, if and only if:

(Q6_T) For each (standard or non-standard) number α:

$$N_+, [\alpha/x] \models \text{(Q6)}.$$

(Q7_T) For all (standard or non-standard) numbers α and β:

$$N_+, [\alpha/x, \beta/y] \models \text{(Q7)}.$$

The case α standard in (Q6$_\top$) and the case α and β standard in (Q7$_\top$) always hold by the definition of '— is an f.n.s. $\mathcal{L}^+$-model'. By Lemmas 5.5.16, 5.5.18 and 5.5.20 the remaining cases hold if and only if we have (a), (b) and (c).

¶ 5.5.24 **Remark** We could have merged Fact 5.5.14(b) and Fact 5.5.14(c) into the following.

> For each (standard or non-standard) α, for each cycle index i, and for each positive integer j:
>
> $$\alpha \times a_i(j) = \oplus_a^{j-1}(\alpha \times a_{i,1}).$$

However, I found it worth highlighting that the case α standard is equivalent to something simpler.

¶ 5.5.25 **Corollary [of Fact 5.5.14]** Suppose $N \models \mathsf{Q}$. Then N is uniquely determined by its $\mathcal{L}^+$-reduct together with the restriction of $\times$ to

$$\{\langle \alpha, a_{i,1}\rangle \;:\; \alpha \text{ standard or non-standard, } i \text{ cycle index}\}.$$

¶ 5.5.26 **Proof** We need to show that this uniquely determines $\times$. The definition of '— is an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model' determines $\times$ on the standard numbers. Given the $\mathcal{L}^+$-reduct and the given restriction of $\times$, the remaining cases are determined by the equations in Fact 5.5.14.

¶ 5.5.27 **Fact** Suppose $N \models \mathsf{Q}$. Then for each cycle index i and for all $a_{i,j}$ and $a_{i,k}$ in A_i:

$$0 \times a_{i,j} = 0 \times a_{i,k}.$$

¶ 5.5.28 **Proof**

$$\begin{aligned} 0 \times a_{i,j} &= 0 \times a_i(j) \\ &= S^{0\times(j-1)}(0 \times a_{i,1}) && \text{(by Fact 5.5.14(b))} \\ &= S^{0\times(k-1)}(0 \times a_{i,1}) \\ &= 0 \times a_i(k) && \text{(ditto)} \\ &= 0 \times a_{i,k}. \end{aligned}$$

¶ 5.5.29 **Fact** Suppose $N \models \mathbf{Q}$. Then for each standard $n > 0$ and for each cycle index i there is a cycle index j such that:

- $n \times a$ is in A_j for each a in A_i.
- $\mu[j]$ divides $n \times \mu[i]$.

¶ 5.5.30 **Proof** Let $n > 0$ be standard and let i be a cycle index. With a proof similar to the proof of Lemma 5.4.20, one may show that it suffices to prove that $n \times a_{i,1}$ is in a cycle of length dividing $n \times \mu[i]$. We have

$$(\dagger) \qquad n \times a_{i,1} = S^{n \times \mu[i]}(n \times a_{i,1})$$

by

$$\begin{aligned} n \times a_{i,1} &= n \times S a_{i,\mu[i]} \\ &= n \times a_i(\mu[i] + 1) \\ &= S^{n \times (\mu[i]+1-1)}(n \times a_{i,1}) \qquad \text{(by Fact 5.5.14(b))} \\ &= S^{n \times \mu[i]}(n \times a_{i,1}). \end{aligned}$$

- By (†) and since $n > 0$, $n \times a_{i,1}$ must be non-standard, say

$$(\ddagger) \qquad n \times a_{i,1} \text{ is in } A_j.$$

- By (†), (‡) and Lemma 5.3.22, $\mu[j]$ divides $n \times \mu[i]$.

¶ 5.5.31 **Fact** Suppose $N \models \mathbf{Q}$. Then for all non-standard a and $a_{i,j}$:

$$a \times a_{i,j} = \oplus_a^{\mu[i]}(a \times a_{i,j}).$$

¶ 5.5.32 **Proof**

$$\begin{aligned} a \times a_{i,j} &= a \times a_i(j) \\ &= a \times a_i(j + \mu[i]) \\ &= \oplus_a^{j+\mu[i]-1}(a \times a_{i,1}) \qquad \text{(by Fact 5.5.14(c))} \\ &= \oplus_a^{\mu[i]} \oplus_a^{j-1} (a \times a_{i,1}) \\ &= \oplus_a^{\mu[i]}(a \times a_{i,j}) \qquad \text{(ditto).} \end{aligned}$$

¶ 5.5.33 **Corollary** Suppose $N \models \mathbf{Q}$. Then $a \times b$ is non-standard for all non-standard a and b.

¶ 5.5.34 **Proof** Suppose $b = a_{i,-}$ for some cycle index i. We then have

$$\begin{aligned} a \times b &= a \times a_{i,-} \\ &= \oplus_a^{\mu[i]}(a \times a_{i,-}) && \text{(by Fact 5.5.31)} \\ &= (\oplus_a^{\mu[i]-1}(a \times a_{i,-})) + a, \end{aligned}$$

which is non-standard by Fact 5.4.18.

¶ 5.5.35 Similar to the corresponding situation in § 5.4, both Facts 5.5.29 and 5.5.31 are in a sense included in Fact 5.5.14. And just as we had an in a sense more practically useful alternative to Fact 5.4.6 (namely, Fact 5.4.24), we here have such an alternative to Fact 5.5.14 (namely, Fact 5.5.36 below).

¶ 5.5.36 **Fact** N is a model of $\mathbf{Q}$ if and only if:

(a) For each non-standard a and for each standard n:

$$a \times n = \oplus_a^n 0.$$

(b) For each standard n and for each non-standard $a_{i,j}$:

$$n \times a_{i,j} = S^{n \times (j-1)}(n \times a_{i,1}).$$

(c) For all non-standard a and $a_{i,j}$:

$$a \times a_{i,j} = \oplus_a^{j-1}(a \times a_{i,1}).$$

(d) For each standard $n > 0$ and for each cycle index i there is a cycle index j such that:

  (1) $n \times a_{i,1}$ is in A_j.

  (2) $\mu[j]$ divides $n \times \mu[i]$.

(e) For each non-standard a and for each cycle index i:

$$a \times a_{i,1} = \oplus_a^{\mu[i]}(a \times a_{i,1}).$$

¶ 5.5.37 **Proof** The only if direction is immediate by Facts 5.5.14, 5.5.29 and 5.5.31.

For the if direction it clearly suffices to prove the respective statements of Facts 5.5.14(b) and 5.5.14(c)—that is:

- The statement of Fact 5.5.14(b):

  For each standard n, for each cycle index i, and for each integer j:
  $$n \times a_i(j) = S^{n\times(j-1)}(n \times a_{i,1}).$$

- The statement of Fact 5.5.14(c):

  For each non-standard a, for each cycle index i, and for each positive integer j:
  $$a \times a_i(j) = \oplus_a^{j-1}(a \times a_{i,1}).$$

- Proof of the statement of Fact 5.5.14(b):

  The case $n = 0$ is taken care of by Fact 5.5.27, which may be proved using (b) similar to how it was proved using Fact 5.5.14(b). Thus let $n > 0$. We have an $a_{i,r}$ and an integer p such that:

  (†) $$a_i(j) = a_{i,r}$$
  (‡) $$j = r + p \times \mu[i].$$

  We then have

  $$\begin{aligned} & n \times a_i(j) \\ &= n \times a_{i,r} && \text{(by (†))} \\ &= S^{n\times(r-1)}(n \times a_{i,1}) && \text{(by (b))} \\ &= S^{n\times(r-1)}S^{n\times p\times\mu[i]}(n \times a_{i,1}) && \text{(by Lemma 5.3.22 and (d))} \\ &= S^{n\times(r+p\times\mu[i]-1)}(n \times a_{i,1}) \\ &= S^{n\times(j-1)}(n \times a_{i,j}) && \text{(by (‡)).} \end{aligned}$$

- Proof of the statement of Fact 5.5.14(b):

  We have an $a_{i,r}$ and a natural number p such that:

  (†) $$a_i(j) = a_{i,r}$$
  (‡) $$j = r + p \times \mu[i].$$

We then have

$$
\begin{aligned}
a \times a_i(j) &= a \times a_{i,r} && \text{(by (†))} \\
&= \oplus_a^{r-1}(a \times a_{i,1}) && \text{(by (c))} \\
&= \oplus_a^{r-1} \oplus_a^{p\times\mu[i]} (a \times a_{i,1}) && \text{(by } p \text{ applications of (e))} \\
&= \oplus_a^{r+p\times\mu[i]-1}(a \times a_{i,1}) && \\
&= \oplus_a^{j-1}(a \times a_{i,1}) && \text{(by (‡))}.
\end{aligned}
$$

¶ 5.5.38 Similarly to how it was easy to extract a recipe (¶ 5.4.28) from Fact 5.4.24 for how to construct—up to isomorphism—any f.n.s. $\mathcal{L}^+$-model of Q^+, Fact 5.5.36 together with Corollary 5.5.33 tell us how to extend that recipe to a recipe for constructing—up to isomorphism—any f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q.

¶ 5.5.39 By Fact 5.5.36 and Corollary 5.5.33, up to isomorphism each f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q may be constructed by following the below instructions for how to turn our arbitrarily chosen f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model N of Q^+ into a concrete model of Q.

(a) Follow the recipe in ¶ 5.4.28 to make the $\mathcal{L}^+$-reduct concrete, and in so doing make sure (d) below may be carried out.

(b) For each cycle index i: choose any (standard or non-standard) number α and set
$$0 \times a_{i,1} := \alpha.$$

(c) For each standard $n > 0$ and for each cycle index i: choose a non-standard a in a cycle of length dividing $n \times \mu[i]$ and set
$$n \times a_{i,1} := a.$$

(d) For each non-standard a and for each cycle index i: choose a non-standard b such that
$$b = \oplus_a^{\mu[i]} b$$
and set
$$a \times a_{i,1} := b.$$

(e) Refer to the equations in Fact 5.5.36 for how to define those remaining multiplications that involve non-standard numbers. (Those multiplications only involving standard numbers are of course as expected—and given by the definition of '— is an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model'.)

¶ 5.5.40 **Remark** Corollary 5.5.33 justifies that the choice in ¶ 5.5.39(d) must be non-standard.

¶ 5.5.41 **Example** The model in Example 5.4.29 was an f.n.s. $\mathcal{L}^{+}$-model of $\mathbf{Q}^{+}$ in the form of a concretization of N_{+}. Following the recipe in ¶ 5.5.39, one may expand that to an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of $\mathbf{Q}$ and end up with the following concretization of N:[*]

- $\mathcal{L}^{+}$-reduct: the model from Example 5.4.29—see ¶ 5.B.4 for a complete explicit definition.
- For n and m standard:

$$n \times m = \text{the ordinary product of } n \text{ and } m.$$

- $a \times n$ for a non-standard and n standard:

$$a \times n = \oplus_a^n 0.$$

- $n \times a_{i,-}$ for n standard, and for $i = 1$ and $i = 2$:

$$n \times a_{i,1} = a_{i,1}$$
$$n \times a_{i,2} = S^n a_{i,1} = a_{i,1} \text{ if } n \text{ even, } a_{i,2} \text{ if } n \text{ odd.}$$

[*] § 5.C provides a Coq formalization which verifies that the provided concretization of N indeed is a model of $\mathbf{Q}$.

– $a \times b$ for a and b non-standard:

$$\begin{aligned} a_{1,1} \times a_{1,1} &= a_{1,1} & a_{1,1} \times a_{2,1} &= a_{1,1} \\ a_{1,2} \times a_{1,1} &= a_{1,1} & a_{1,2} \times a_{2,1} &= a_{1,1} \\ a_{2,1} \times a_{1,1} &= a_{2,1} & a_{2,1} \times a_{2,1} &= a_{2,1} \\ a_{2,2} \times a_{1,1} &= a_{2,2} & a_{2,2} \times a_{2,1} &= a_{2,2} \\ a_{1,1} \times a_{1,2} &= a_{1,1} & a_{1,1} \times a_{2,2} &= a_{1,1} \\ a_{1,2} \times a_{1,2} &= a_{1,2} & a_{1,2} \times a_{2,2} &= a_{1,2} \\ a_{2,1} \times a_{1,2} &= a_{2,1} & a_{2,1} \times a_{2,2} &= a_{2,1} \\ a_{2,2} \times a_{1,2} &= a_{2,2} & a_{2,2} \times a_{2,2} &= a_{2,2}. \end{aligned}$$

¶ 5.5.42 **Facts**

(a) There is a commutative associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q.

(b) There is a commutative non-associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q.

(c) There is a non-commutative associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q.

(d) There is a non-commutative non-associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^{\mathsf{Q}}$-model of Q.

¶ 5.5.43 **Proofs** Each f.n.s. $\mathcal{L}$-model of Q is isomorphic to one constructed following the recipe in ¶ 5.5.39. Thus consider a thus constructed f.n.s. $\mathcal{L}$-model of Q. ¶ 5.5.39(a) tells us that the chosen $\mathcal{L}^+$-reduct modeling Q^+ must make ¶ 5.5.39(d) possible to carry out. Thus to prove the facts, for each of (a)–(d) we construct a suitable $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is impossible to carry out.* The actual constructions are not that interesting. I defer those to § 5.A.

¶ 5.5.44 **Fact** There are uncountably many non-isomorphic f.n.s. models of Q.

* The non-standard part of each of these models were found by automated search procedures. These search procedures are developed and described in Ch. 6, where I also provide Python implementations of them.

¶ 5.5.45 **Proof** Consider concretizing N into an f.n.s. model of **Q** by following the recipe in ¶ 5.5.39. To carry out ¶ 5.5.39(a) we choose the following $\mathcal{L}^+$-reduct.

– We choose the cycle structure:

$$A \coloneqq A_1 \coloneqq \{a_{1,1},\ a_{1,2}\}.$$

– For each natural number n we choose:

$$n + a_{1,1} \coloneqq a_{1,1}.$$

– We choose:

$$a_{1,1} + a_{1,1} \coloneqq a_{1,1}$$
$$a_{1,2} + a_{1,1} \coloneqq a_{1,2}.$$

Under the constraint that we should have a model of $\mathbf{Q}^+$, these choices uniquely determines the $\mathcal{L}^+$reduct—for this I refer skeptic readers to the recipe for concretizing N_+ into a model of $\mathbf{Q}^+$ (¶ 5.4.28).

To carry out ¶¶ 5.5.39(b) and 5.5.39(d), we choose:

(†) $$0 \times a_{1,1} \coloneqq 0$$

(‡) $$a_{1,1} \times a_{1,1} \coloneqq a_{1,1} \eqqcolon a_{2,1} \times a_{1,1}.$$

It remains to carry out ¶¶ 5.5.39(c) and 5.5.39(e). For ¶ 5.5.39(c) we may, for each standard $n > 0$, choose either $n \times a_{1,1} \coloneqq a_{1,1}$ or $n \times a_{1,1} \coloneqq a_{1,2}$, and no matter our choices we end up with a model of **Q** after carrying out—in the only way possible—¶ 5.5.39(e). Thus there is a bijection between

$$\mathbb{N} \to \{a_{1,1},\ a_{1,2}\}$$

and the set C of N-concretizations such that for each concretization c in C: c has the given $\mathcal{L}^+$-reduct, $c \models \mathbf{Q}$, and c satisfies (†) and (‡). Since distinct such concretizations are non-isomorphic, we thus have uncountably many non-isomorphic f.n.s. models of **Q**.

¶ 5.5.46 **Open problem?** Is there a recursively enumerable set R of recursive presentations of f.n.s. models of **Q** such that, up to isomorphism, each recursive f.n.s. model of **Q** has a representation in R?

§ 5.A Some proofs

¶ 5.A.1 **Lemmas 5.3.15 (restated)** For each non-standard $a_{i,j}$ and for each integer n:

(a) $S^n a_{i,j} = a_i(j+n)$

(b) $P^n a_{i,j} = a_i(j-n)$

(c) $a_{i,j} = S^n a_i(j-n)$

(d) $a_{i,j} = P^n a_i(j+n)$

(e) $a_{i,j} = S^{n\times\mu[i]} a_{i,j}$

(f) $a_{i,j} = P^{n\times\mu[i]} a_{i,j}$.

¶ 5.A.2 **More detailed proofs than Proofs 5.3.16**

- For $n \geq 0$, (a) and (b) are both straightforwardly provable by induction, and then for $n < 0$ they follow from each other by their definitions (Definitions 5.3.12(a)).

- For (c), by definition of '$a\_$' we have a k such that:

  (†) $$a_i(j-n) = a_{i,k}$$
  (‡) $$j - n \equiv k \mod \mu[i].$$

  Then:

  $$\begin{aligned} a_i(j) &= a_i(j-n+n) \\ &= a_i(k+n) && \text{(by (‡) and the definition of '}a\_\text{')} \\ &= S^n a_{i,k} && \text{(by (a))} \\ &= S^n a_i(j-n) && \text{(by (†)).} \end{aligned}$$

  One can prove (d) similarly.

- (e) and (f) follow from the definition of '$a\_$', together with (a) and (b), respectively.

¶ 5.A.3 **Lemmas 5.3.19 (restated)** For each non-standard a and for all integers k and m:

(a) $S^k S^m a = S^{k+m} a$

(b) $P^k P^m a = P^{k+m} a$.

¶ 5.A.4 **Proofs**

(a) We have

(†) $S^k S^m a_{i,j} = S^k a_i(j+m)$ (by Lemma 5.3.15(a)).

By definition of '$a\_$', we have an n such that $a_{i,n} \in A_i$ and

(‡) $a_i(j+m) = a_{i,n}$

(⊠) $n \equiv j+m \mod \mu[i]$.

Then

$$
\begin{aligned}
S^k S^m a_{i,j} &= S^k a_i(j+m) && \text{(by (†))}\\
&= S^k a_{i,n} && \text{(by (‡))}\\
&= a_i(n+k) && \text{(by Lemma 5.3.15(a)).}\\
&= a_i(j+m+k) && \text{(by (⊠) and definition of '}a\_\text{')}\\
&= S^{k+m} a_{i,j} && \text{(by Lemma 5.3.15(a)).}
\end{aligned}
$$

(b) Similar to 5.1.4(a), using Lemma 5.3.15(b) instead of Lemma 5.3.15(a).

¶ 5.A.5 **Lemma 5.3.22 (restated)** For each non-standard $a_{i,j}$ and for each integer n, we have

$$S^n a_{i,j} = a_{i,j}$$

and

$$P^n a_{i,j} = a_{i,j}$$

if $\mu[i]$ divides n—otherwise we have neither.

¶ 5.A.6 **Proof** We have

$$S^n a_{i,j} = a_i(j+n)$$

and, by definition of '$a\_$', we have

$$a_i(j+n) = a_{i,j}$$

if and only if

$$j + n \equiv j \mod \mu[i],$$

that is, if and only if

$$n \equiv 0 \mod \mu[i],$$

that is, if and only if $\mu[i]$ divides n. Similarly, we have $P^n a_{i,j} = a_{i,j}$ if and only if n divides $\mu[i]$.

¶ 5.A.7 **Facts 5.5.42 (restated)**

(a) There is a commutative associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^\mathsf{Q}$-model of Q.

(b) There is a commutative non-associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^\mathsf{Q}$-model of Q.

(c) There is a non-commutative associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^\mathsf{Q}$-model of Q.

(d) There is a non-commutative non-associative f.n.s. $\mathcal{L}^+$-model of Q^+ that cannot be expanded to an f.n.s. $\mathcal{L}^\mathsf{Q}$-model of Q.

¶ 5.A.8 **Proofs** Each f.n.s. $\mathcal{L}$-model of Q is isomorphic to one constructed following the recipe in ¶ 5.5.39. Thus consider a thus constructed f.n.s. $\mathcal{L}$-model of Q. ¶ 5.5.39(a) tells us that the chosen $\mathcal{L}^+$-reduct modeling Q^+ must make ¶ 5.5.39(d) possible to carry out. Thus to prove the facts, for each of (a)–(d) we construct a suitable $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is impossible to carry out.*

* The non-standard part of each of these models were found by automated search procedures. These search procedures are developed and described in Ch. 6, where I also provide Python implementations of them.

(a) We have the following concretization of N_+:

$$\begin{aligned} A &\coloneqq A_1 + A_2 \\ A_1 &\coloneqq \{a_{1,1}\} \\ A_2 &\coloneqq \{a_{2,1}\} \end{aligned}$$

$$\begin{aligned} a_{1,1} + n &\coloneqq a_{1,1} & a_{2,1} + n &\coloneqq a_{2,1} & &(n \text{ standard}) \\ n + a_{1,1} &\coloneqq a_{1,1} & n + a_{2,1} &\coloneqq a_{2,1} & &(n \text{ standard}) \\ a_{1,1} + a_{1,1} &\coloneqq a_{1,1} & a_{1,1} + a_{2,1} &\coloneqq a_{2,1} \\ a_{2,1} + a_{1,1} &\coloneqq a_{2,1} & a_{2,1} + a_{2,1} &\coloneqq a_{1,1}. \end{aligned}$$

The output when running the Python script in § 5.D verifies that this is a commutative and associative $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is not possible to carry out.

(b) We have the following concretization of N_+:

$$\begin{aligned} A &\coloneqq A_1 + A_2 \\ A_1 &\coloneqq \{a_{1,1}\} \\ A_2 &\coloneqq \{a_{2,1}\} \end{aligned}$$

$$\begin{aligned} a_{1,1} + n &\coloneqq a_{1,1} & a_{2,1} + n &\coloneqq a_{2,1} & &(n \text{ standard}) \\ n + a_{1,1} &\coloneqq a_{1,1} & n + a_{2,1} &\coloneqq a_{2,1} & &(n \text{ standard}) \\ a_{1,1} + a_{1,1} &\coloneqq a_{2,1} & a_{1,1} + a_{2,1} &\coloneqq a_{1,1} \\ a_{2,1} + a_{1,1} &\coloneqq a_{1,1} & a_{2,1} + a_{2,1} &\coloneqq a_{1,1}. \end{aligned}$$

The output when running the Python script in § 5.D verifies that this is a commutative non-associative $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is not possible to carry out.

(c) We have the following concretization of N_+:

$$\begin{aligned} A &\coloneqq A_1 + A_2 + A_3 \\ A_1 &\coloneqq \{a_{1,1},\ a_{1,2}\} \\ A_2 &\coloneqq \{a_{2,1}\} \\ A_3 &\coloneqq \{a_{3,1}\} \\ a_{1,1} + n &\coloneqq a_{1,1} \text{ if } n \text{ even, } a_{1,2} \text{ if } n \text{ odd} \\ &\eqqcolon n + a_{1,1} \qquad\qquad (n \text{ standard}) \end{aligned}$$

$$
\begin{aligned}
a_{1,2} + n &\coloneqq a_{1,2} \text{ if } n \text{ even, } a_{1,1} \text{ if } n \text{ odd} \\
&\eqqcolon n + a_{1,2} && (n \text{ standard}) \\
a_{2,1} + n &\coloneqq a_{2,1} \eqqcolon n + a_{2,1} && (n \text{ standard}) \\
a_{3,1} + n &\coloneqq a_{3,1} \eqqcolon n + a_{3,1} && (n \text{ standard}) \\
a_{1,1} + a_{1,1} &\coloneqq a_{1,1} \\
a_{1,2} + a_{1,1} &\coloneqq a_{1,1} \\
a_{2,1} + a_{1,1} &\coloneqq a_{2,1} \\
a_{3,1} + a_{1,1} &\coloneqq a_{3,1} \\
a_{1,1} + a_{1,2} &\coloneqq a_{1,2} \\
a_{1,2} + a_{1,2} &\coloneqq a_{1,2} \\
a_{2,1} + a_{1,2} &\coloneqq a_{2,1} \\
a_{3,1} + a_{1,2} &\coloneqq a_{3,1} \\
a_{1,1} + a_{2,1} &\coloneqq a_{2,1} \\
a_{1,2} + a_{2,1} &\coloneqq a_{2,1} \\
a_{2,1} + a_{2,1} &\coloneqq a_{2,1} \\
a_{3,1} + a_{2,1} &\coloneqq a_{3,1} \\
a_{1,1} + a_{3,1} &\coloneqq a_{3,1} \\
a_{1,2} + a_{3,1} &\coloneqq a_{3,1} \\
a_{2,1} + a_{3,1} &\coloneqq a_{3,1} \\
a_{3,1} + a_{3,1} &\coloneqq a_{2,1}.
\end{aligned}
$$

The output when running the Python script in § 5.D verifies that this is a non-commutative associative $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is not possible to carry out.

(d) We have the following concretization of N_+:

$$
\begin{aligned}
A &\coloneqq A_1 \\
A_1 &\coloneqq \{a_{1,1},\ a_{1,2}\} \\
a_{1,1} + n &\coloneqq a_{1,1} \text{ if } n \text{ even, } a_{1,2} \text{ if } n \text{ odd} \\
&\eqqcolon n + a_{1,1} && (n \text{ standard}) \\
a_{1,2} + n &\coloneqq a_{1,2} \text{ if } n \text{ even, } a_{1,1} \text{ if } n \text{ odd} \\
&\eqqcolon n + a_{1,2} && (n \text{ standard}) \\
a_{1,1} + a_{1,1} &\coloneqq a_{1,2}
\end{aligned}
$$

$$a_{1,2} + a_{1,1} \coloneqq a_{1,2}$$
$$a_{1,1} + a_{1,2} \coloneqq a_{1,1}$$
$$a_{1,2} + a_{1,2} \coloneqq a_{1,1}.$$

The output when running the Python script in § 5.D verifies that this is a non-commutative non-associative $\mathcal{L}^+$-model of Q^+ for which ¶ 5.5.39(d) is not possible to carry out.

§ 5.B The complete and explicit definition of the model from Example 5.4.29

¶ 5.B.1 We compute those additions not explicitly defined in Example 5.4.29, continuing where we left off in the model construction recipe (¶ 5.4.28).

¶ 5.B.2 Recall that the model is a concretization of N_+. We recall what we had explicitly defined so far.

– $\mathcal{L}^{\mathrm{p}}$-reduct:

$$\begin{aligned} A &= A_1 + A_2 \\ A_1 &= \{a_{1,1}, a_{1,2}\} \\ A_2 &= \{a_{2,1}, a_{2,2}\} \\ Sa_{1,1} &= a_{1,2} \\ Sa_{1,2} &= a_{1,1} \\ Sa_{2,1} &= a_{2,2} \\ Sa_{2,2} &= a_{2,1}. \end{aligned}$$

– $n + a_{i,-}$ for n standard and for $i = 1$ and $i = 2$:

(†) $n + a_{i,1} = S^n a_{i,1} = a_{i,1} + n = a_{i,1}$ if n even, $a_{i,2}$ if n odd.

– $a + a_{i,-}$ for a non-standard and for $i = 1$ and $i = 2$:

(111) $$a_{1,1} + a_{1,1} = a_{1,1}$$
(121) $$a_{1,2} + a_{1,1} = a_{1,2}$$
(211) $$a_{2,1} + a_{1,1} = a_{2,1}$$
(221) $$a_{2,2} + a_{1,1} = a_{2,2}$$
(112) $$a_{1,1} + a_{2,1} = a_{2,2}$$
(122) $$a_{1,2} + a_{2,1} = a_{2,2}$$
(212) $$a_{2,1} + a_{2,1} = a_{2,1}$$
(222) $$a_{2,2} + a_{2,1} = a_{2,1}.$$

¶ 5.B.3 Following the recipe (¶ 5.4.28), we use the equations in Fact 5.4.24 to compute the remaining additions that involve non-standard numbers. (The additions that involve only standard numbers are of course defined as usual.)

– $a_{i,-} + n$ for $i = 1$ and $i = 2$ and for n standard:

$$\begin{aligned} a_{i,1} + n &= S^n a_{i,1} && \text{(by Fact 5.4.24(a))} \\ &= \mathrm{a}_{i,1} \text{ if } n \text{ even, } a_{i,2} \text{ if } n \text{ odd} \end{aligned}$$

$$\begin{aligned} a_{i,2} + n &= S^n a_{i,2} && \text{(ditto)} \\ &= a_{i,2} \text{ if } n \text{ even, } a_{i,1} \text{ if } n \text{ odd.} \end{aligned}$$

– $n + a_{i,2}$ for n standard and for $i = 1$ and $i = 2$:

$$\begin{aligned} n + a_{i,2} &= S^{2-1}(n + a_{i,1}) && \text{(by Fact 5.4.24(b))} \\ &= S(n + a_{i,1}) \\ &= SS^n a_{i,1} && \text{(by (†))} \\ &= S^n a_{i,2} \\ &= a_{i,2} \text{ if } n \text{ even, } a_{i,1} \text{ if } n \text{ odd.} \end{aligned}$$

– $a + a_{i,2}$ for a non-standard and for $i = 1$ and $i = 2$:

We have

(‡) $$a_{k,l} + a_{i,2} = S(a_{k,l} + a_{i,1})$$

by

$$\begin{aligned} a_{k,l} + a_{i,2} &= S^{2-1}(a_{k,l} + a_{i,1}) && \text{(by Fact 5.4.24(b))} \\ &= S(a_{k,l} + a_{i,1}). \end{aligned}$$

Thus:

$$\begin{aligned} a_{1,1} + a_{1,2} &= S(a_{1,1} + a_{1,1}) && \text{(by (‡))} \\ &= Sa_{1,1} && \text{(by (111)} \\ &= a_{1,2} \end{aligned}$$

$$\begin{aligned} a_{1,2} + a_{1,2} &= S(a_{1,2} + a_{1,1}) && \text{(by (‡))} \\ &= Sa_{1,2} && \text{(by (121))} \\ &= a_{1,1} \end{aligned}$$

$$\begin{aligned} a_{2,1} + a_{1,2} &= S(a_{2,1} + a_{1,1}) && \text{(by (‡))} \\ &= Sa_{2,1} && \text{(by (211))} \\ &= a_{2,2} \end{aligned}$$

$$\begin{aligned} a_{2,2} + a_{1,2} &= S(a_{2,2} + a_{1,1}) && \text{(by (‡))} \\ &= Sa_{2,2} && \text{(by (221))} \\ &= a_{2,1} \end{aligned}$$

$$\begin{aligned} a_{1,1} + a_{2,2} &= S(a_{1,1} + a_{2,1}) && \text{(by (‡))} \\ &= Sa_{2,2} && \text{(by (112))} \\ &= a_{2,1} \end{aligned}$$

$$\begin{aligned} a_{1,2} + a_{2,2} &= S(a_{1,2} + a_{2,1}) && \text{(by (‡))} \\ &= Sa_{2,2} && \text{(by (122))} \\ &= a_{2,1} \end{aligned}$$

$$\begin{aligned} a_{2,1} + a_{2,2} &= S(a_{2,1} + a_{2,1}) && \text{(by (‡))} \\ &= Sa_{2,1} && \text{(by (212))} \end{aligned}$$

$$= a_{2,2}$$

$$\begin{aligned} a_{2,2} + a_{2,2} &= S(a_{2,2} + a_{2,1}) && \text{(by (‡))} \\ &= Sa_{2,1} && \text{(by (222))} \\ &= a_{2,2}. \end{aligned}$$

¶ 5.B.4 All in all, we have an f.n.s. $\mathcal{L}^+$-model of Q^+ in the following concretization of N_+:

– $\mathcal{L}^{\mathrm{p}}$-reduct:

$$\begin{aligned} A &= A_1 + A_2 \\ A_1 &= \{a_{1,1}, a_{1,2}\} \\ A_2 &= \{a_{2,1}, a_{2,2}\} \\ Sa_{1,1} &= a_{1,2} \\ Sa_{1,2} &= a_{1,1} \\ Sa_{2,1} &= a_{2,2} \\ Sa_{2,2} &= a_{2,1}. \end{aligned}$$

– For n and m standard:

$$n + m = \text{the ordinary sum of } n \text{ and } m.$$

$n + a_{i,1}$ for n standard and for $i = 1$ and $i = 2$:

$$\begin{aligned} n + a_{i,1} &= a_{i,1} + n = S^n a_{i,1} = a_{i,1} \text{ if } n \text{ even, } a_{i,2} \text{ if } n \text{ odd} \\ n + a_{i,2} &= a_{i,2} + n = S^n a_{i,2} = a_{i,2} \text{ if } n \text{ even, } a_{i,1} \text{ if } n \text{ odd.} \end{aligned}$$

– $a + b$ for a and b non-standard:

$$\begin{aligned} a_{1,1} + a_{1,1} &= a_{1,1} & a_{1,1} + a_{2,1} &= a_{2,2} \\ a_{1,2} + a_{1,1} &= a_{1,2} & a_{1,2} + a_{2,1} &= a_{2,2} \\ a_{2,1} + a_{1,1} &= a_{2,1} & a_{2,1} + a_{2,1} &= a_{2,1} \\ a_{2,2} + a_{1,1} &= a_{2,2} & a_{2,2} + a_{2,1} &= a_{2,1} \\ a_{1,1} + a_{1,2} &= a_{1,2} & a_{1,1} + a_{2,2} &= a_{2,1} \\ a_{1,2} + a_{1,2} &= a_{1,1} & a_{1,2} + a_{2,2} &= a_{2,1} \\ a_{2,1} + a_{1,2} &= a_{2,2} & a_{2,1} + a_{2,2} &= a_{2,2} \\ a_{2,2} + a_{1,2} &= a_{2,1} & a_{2,2} + a_{2,2} &= a_{2,2}. \end{aligned}$$

§ 5.C A Coq formalization verifying that Example 5.5.41 provides a model of Robinson arithmetic

The following Coq source type checks with Coq 8.20.1.

```
Require Import Arith.

Definition models_Q_p
  (M : Type) (O_M : M) (S_M : M -> M)
  :
  Prop
  :=
    (forall x, S_M x <> O_M)                          (* (Q1) *)
    /\
    (forall x y, S_M x = S_M y -> x = y)              (* (Q2) *)
    /\
    (forall x, x = O_M \/ exists y, x = S_M y). (* (Q3) *)

Definition models_Q_add
  (M : Type) (O_M : M) (S_M : M -> M) (add_M : M -> M -> M)
  :
  Prop
  :=
    models_Q_p M O_M S_M
    /\
    (forall x, add_M x O_M = x)                          (* (Q4) *)
    /\
    (forall x y, add_M x (S_M y) = S_M (add_M x y)). (* (Q5) *)

Definition models_Q
  (M      : Type)
  (O_M    : M)
  (S_M    : M -> M)
  (add_M  : M -> M -> M)
  (mult_M : M -> M -> M)
```

```
  :
  Prop
  :=
    models_Q_add M O_M S_M add_M
    /\
    (forall x, mult_M x O_M = O_M)
        (* (Q6) *)
    /\
    (forall x y, mult_M x (S_M y) = add_M (mult_M x y) x).
        (* (Q7) *)

Fact nat_models_Q : models_Q nat 0 S plus mult.
Proof.
  unfold models_Q. repeat split; auto.
  induction x as [| x IH].
  - left. reflexivity.
  - destruct IH as [IH1 | IH2]; right; eauto.
Qed.

Inductive A : Type :=
| a11 : A
| a12 : A
| a21 : A
| a22 : A.

Definition S_A (a : A) := match a with
| a11 => a12
| a12 => a11
| a21 => a22
| a22 => a21
end.

Definition fns_N : Type := nat + A.

Definition O_N : fns_N := inl 0.

Definition S_N (a : fns_N) : fns_N := match a with
| inl n => inl (S n)
| inr a => inr (S_A a)
```

```
end.

Fact fns_N_with_S_N_models_Q_p : models_Q_p fns_N O_N S_N.
Proof.
  unfold models_Q_p. repeat split.
  - intro α. destruct α as [n | a].
    + simpl. unfold O_N. injection. intros H2. inversion H2.
    + unfold O_N. destruct a; simpl; intro H; inversion H.
  - intros α β H. destruct α, β.
    + simpl in H. injection H. auto.
    + repeat unfold S_N in H. destruct a; simpl; inversion H.
    + repeat unfold S_N in H. destruct a; simpl; inversion H.
    + destruct a, a0; auto.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
      * simpl in H. inversion H.
  - intros α. destruct α as [n | a].
    + destruct n as [|n].
      * left. auto.
      * right. exists (inl n). reflexivity.
    + right. destruct a.
      * exists (inr a12); reflexivity.
      * exists (inr a11); reflexivity.
      * exists (inr a22); reflexivity.
      * exists (inr a21); reflexivity.
Qed.

Definition add_N_ns_std (a : A) (n : nat) :=
match a with
| a11 => if Nat.even n then a11 else a12
| a12 => if Nat.even n then a12 else a11
```

```
| a21 => if Nat.even n then a21 else a22
| a22 => if Nat.even n then a22 else a21
end.

Definition add_N_std_ns (n : nat) (a : A) : A :=
    add_N_ns_std a n.

Definition add_N_ns_ns (a b : A) :=
match a, b with
| a11, a11 => a11
| a11, a12 => a12
| a11, a21 => a22
| a11, a22 => a21
| a12, a11 => a12
| a12, a12 => a11
| a12, a21 => a22
| a12, a22 => a21
| a21, a11 => a21
| a21, a12 => a22
| a21, a21 => a21
| a21, a22 => a22
| a22, a11 => a22
| a22, a12 => a21
| a22, a21 => a21
| a22, a22 => a22
end.

Definition add_N (α β : fns_N) : fns_N :=
match α, β with
| inl n, inl m => inl (n+m)
| inl n, inr a => inr (add_N_std_ns n a)
| inr a, inl n => inr (add_N_ns_std a n)
| inr a, inr b => inr (add_N_ns_ns a b)
end.

Lemma fns_N_with_S_N_add_N_models_Q4 : forall (α : fns_N),
    add_N α (inl 0) = α.
Proof.
  destruct α as [n | a].
```

```
  + simpl. rewrite <- plus_n_O. reflexivity.
  + simpl. destruct a; reflexivity.
Qed.

Lemma fns_N_with_S_N_add_N_models_Q5_ns_std
  : forall (a : A) (n : nat), add_N_ns_std a (S n) = S_A
      (add_N_ns_std a n).
Proof.
  intros a n.
  unfold add_N_ns_std.
  remember (Nat.even    n ) as   n_even eqn:eq_n_even.
  remember (Nat.even (S n)) as S_n_even eqn:eq_S_n_even.
  destruct n_even; destruct S_n_even.
  - absurd (true = Nat.even (S n)).
    + symmetry in eq_n_even, eq_S_n_even.
      rewrite Nat.even_spec in eq_n_even, eq_S_n_even.
      rewrite Nat.Even_succ in eq_S_n_even.
      apply (Nat.Even_Odd_False n eq_n_even) in
          eq_S_n_even.
      auto.
    + auto.
  - destruct a; reflexivity.
  - destruct a; reflexivity.
  - absurd (false = Nat.even (S n)).
    + symmetry in eq_n_even, eq_S_n_even.
      rewrite <- eq_S_n_even in eq_n_even.
      rewrite Nat.even_succ in eq_n_even, eq_S_n_even.
      rewrite <- Nat.negb_odd in eq_n_even.
      rewrite eq_S_n_even in eq_n_even.
      simpl in eq_n_even.
      inversion eq_n_even.
    + auto.
Qed.

Fact fns_N_with_S_N_add_N_models_Q_add : models_Q_add
    fns_N O_N S_N add_N.
Proof.
  unfold models_Q_add.
  split; [apply fns_N_with_S_N_models_Q_p | ].
```

```
  split.
  - simpl. unfold O_N. apply fns_N_with_S_N_add_N_models_Q4.
  - intros α β.
    + destruct α as [n | a]; [destruct β as [m | a] | destruct β as [n | b]].
      * simpl. rewrite <- plus_n_Sm. reflexivity.
      * simpl. destruct a; simpl; destruct (Nat.even n); reflexivity.
      * simpl. rewrite <- fns_N_with_S_N_add_N_models_Q5_ns_std. reflexivity.
      * simpl. unfold add_N_ns_ns. destruct a; destruct b; reflexivity.
Qed.

Fixpoint it_add_right_N (β α : fns_N) (n : nat) : fns_N :=
match n with
| 0   => α
| S n => add_N (it_add_right_N β α n) β
end.

Definition mult_N_ns_std (a : A) (n : nat) : fns_N :=
  it_add_right_N (inr a) (inl 0) n.

Lemma fns_N_with_S_N_add_N_mult_N_models_ns
  : forall (a : A), mult_N_ns_std a 0 = inl 0.
Proof. intro a. unfold mult_N_ns_std. reflexivity. Qed.

Lemma mult_N_a11_fixpoint : forall (n : nat), mult_N_ns_std a11 (S n) = inr a11.
Proof.
  intro n. induction n as [|n IHn].
  - reflexivity.
  - change
      (mult_N_ns_std a11 (S (S n)))
    with
      (add_N (mult_N_ns_std a11 (S n)) (inr a11)).
    rewrite IHn.
```

```
    reflexivity.
Qed.

Lemma mult_N_a21_fixpoint : forall (n : nat),
    mult_N_ns_std a21 (S n) = inr a21.
Proof.
  intro n. induction n as [|n IHn].
  - reflexivity.
  - change
      (mult_N_ns_std a21 (S (S n)))
    with
      (add_N (mult_N_ns_std a21 (S n)) (inr a21)).
    rewrite IHn.
    reflexivity.
Qed.

Lemma mult_N_a22_fixpoint : forall (n : nat),
    mult_N_ns_std a22 (S n) = inr a22.
Proof.
  intro n. induction n as [|n IHn].
  - reflexivity.
  - change
      (mult_N_ns_std a22 (S (S n)))
    with
      (add_N (mult_N_ns_std a22 (S n)) (inr a22)).
    rewrite IHn.
    reflexivity.
Qed.

Lemma mult_N_a12_even_odd
  : forall n : nat,
    (Nat.Even (S n) -> mult_N_ns_std a12 (S n) = inr a11)
    /\
    (Nat.Odd  (S n) -> mult_N_ns_std a12 (S n) = inr a12).
Proof.
  intro n. induction n as [|n IHn].
  - split.
    + intros H_even_1.
      exfalso.
```

```
      rewrite <- Nat.even_spec in H_even_1.
      rewrite Nat.even_1 in H_even_1.
      inversion H_even_1.
    + reflexivity.
  - split.
    + intros H_even_SSn.
      destruct IHn as [_ IHn].
      rewrite Nat.Even_succ in H_even_SSn.
      specialize (IHn H_even_SSn).
      change
        (mult_N_ns_std a12 (S (S n)))
      with
        (add_N (mult_N_ns_std a12 (S n)) (inr a12)).
      rewrite IHn.
      simpl.
      reflexivity.
    + intros H_odd_SSn.
      destruct IHn as [IHn _].
      rewrite Nat.Odd_succ in H_odd_SSn.
      specialize (IHn H_odd_SSn).
      change
        (mult_N_ns_std a12 (S (S n)))
      with
        (add_N (mult_N_ns_std a12 (S n)) (inr a12)).
      rewrite IHn.
      simpl.
      reflexivity.
Qed.

Lemma mult_N_a12_even
  : forall n : nat,
     n <> 0 -> Nat.Even n -> mult_N_ns_std a12 n = inr a11.
Proof.
  intro n. destruct n as [|n].
  - intro H_0_neq_0. exfalso. apply H_0_neq_0. reflexivity.
  - intros _ H_even_Sn.
    pose (mult_N_a12_even_odd n) as H. destruct H as [H _].
    apply (H H_even_Sn).
Qed.
```

```
Lemma mult_N_a12_odd
  : forall n : nat, Nat.Odd n -> mult_N_ns_std a12 n = inr
     a12.
Proof.
  intro n. destruct n as [|n].
  - intro H_odd_0.
    rewrite <- Nat.odd_spec in H_odd_0.
    rewrite Nat.odd_0 in H_odd_0.
    inversion H_odd_0.
  - intros H_odd_Sn.
    pose (mult_N_a12_even_odd n) as H. destruct H as [_ H].
    apply (H H_odd_Sn).
Qed.

Lemma Sn_neq_0 : forall n : nat, S n <> 0.
  intros n. symmetry. apply Nat.neq_0_succ.
Qed.

Lemma fns_N_with_S_N_add_N_mult_N_models_Q7_ns_std
  : forall (a : A) (n : nat),
      mult_N_ns_std a (S n) = add_N (mult_N_ns_std a n)
          (inr a).
Proof.
  intro a. destruct a.
  - destruct n as [|n].
    + reflexivity.
    + repeat rewrite mult_N_a11_fixpoint. reflexivity.
  - intro n.
    destruct (Nat.Even_or_Odd (S n)) as [H_even_Sn |
        H_odd_Sn].
    + rewrite (mult_N_a12_even (S n) (Sn_neq_0 n)
        H_even_Sn).
      rewrite Nat.Even_succ in H_even_Sn.
      rewrite (mult_N_a12_odd n H_even_Sn).
      simpl.
      reflexivity.
    + rewrite (mult_N_a12_odd (S n) H_odd_Sn).
      destruct n as [|n].
```

```
      * simpl. reflexivity.
      * rewrite Nat.Odd_succ in H_odd_Sn.
        rewrite (mult_N_a12_even (S n) (Sn_neq_0 n)
            H_odd_Sn).
        simpl.
        reflexivity.
  - destruct n as [|n].
    + reflexivity.
    + repeat rewrite mult_N_a21_fixpoint. reflexivity.
  - destruct n as [|n].
    + reflexivity.
    + repeat rewrite mult_N_a22_fixpoint. reflexivity.
Qed.

Definition mult_N_std_ns (n : nat) (a : A) : A :=
match a with
| a11 => a11
| a21 => a21
| a12 => if Nat.even n then a11 else a12
| a22 => if Nat.even n then a21 else a22
end.

Definition mult_N_ns_ns (a b : A) : A :=
match a, b with
| a11, a11 => a11
| a12, a11 => a11
| a21, a11 => a21
| a22, a11 => a22
| a11, a12 => a11
| a12, a12 => a12
| a21, a12 => a21
| a22, a12 => a22
| a11, a21 => a11
| a12, a21 => a11
| a21, a21 => a21
| a22, a21 => a22
| a11, a22 => a11
| a12, a22 => a12
| a21, a22 => a21
```

```
| a22, a22 => a22
end.

Lemma fns_N_with_S_N_add_N_mult_N_models_Q7_ns_ns
  : forall (a b : A), mult_N_ns_ns a (S_A b) = add_N_ns_ns (mult_N_ns_ns a b) a.
Proof.
  intros a b; destruct a; destruct b; simpl; reflexivity.
Qed.

Definition mult_N (α β : fns_N) : fns_N :=
match α, β with
| inl n, inl m => inl (n*m)
| inl n, inr a => inr (mult_N_std_ns n a)
| inr a, inl n =>     (mult_N_ns_std a n)
| inr a, inr b => inr (mult_N_ns_ns a b)
end.

Fact fns_N_with_S_N_add_N_mult_N_models_Q
  : models_Q fns_N O_N S_N add_N mult_N.
Proof.
  unfold models_Q. split; [| split].
  - apply fns_N_with_S_N_add_N_models_Q_add.
  - intro α. destruct α as [n | a].
    + unfold mult_N. simpl. rewrite <- mult_n_O. reflexivity.
    + destruct a;
        unfold mult_N; simpl; unfold mult_N_ns_std; simpl; reflexivity.
  - intros α β.
    destruct α as [n | a]; [destruct β as [m | a] | destruct β as [n | b]].
    + unfold add_N, mult_N. simpl. rewrite <- mult_n_Sm. reflexivity.
    + destruct a.
      * reflexivity.
      * simpl. destruct (Nat.Even_or_Odd n) as [n_even | n_odd].
        {
```

```
          rewrite <- Nat.even_spec in n_even.
          rewrite n_even. simpl. rewrite n_even.
          reflexivity.
        }
        {
          rewrite <- Nat.odd_spec in n_odd.
          set (Nat.negb_odd n) as n_not_even.
         rewrite n_odd in n_not_even. simpl in n_not_even.
          rewrite <- n_not_even. simpl.
          rewrite <- n_not_even. reflexivity.
        }
      * reflexivity.
      * simpl. destruct (Nat.Even_or_Odd n) as [n_even |
          n_odd].
        {
          rewrite <- Nat.even_spec in n_even.
          rewrite n_even. simpl. rewrite n_even.
          reflexivity.
        }
        {
          rewrite <- Nat.odd_spec in n_odd.
          set (Nat.negb_odd n) as n_not_even.
         rewrite n_odd in n_not_even. simpl in n_not_even.
          rewrite <- n_not_even. simpl.
          rewrite <- n_not_even. reflexivity.
        }
    + apply fns_N_with_S_N_add_N_mult_N_models_Q7_ns_std.
    + simpl. rewrite
        fns_N_with_S_N_add_N_mult_N_models_Q7_ns_ns.
        reflexivity.
Qed.
```

§ 5.D Source of Python script referenced in Proofs 5.5.43, and the output from running it

§ 5.D.1 Source

```
#! /usr/bin/env python3.13

# IMPORTS

import dataclasses

from itertools import product
from typing import Final as F, NewType, Self, TypeAlias



# NEWTYPES T_CORRECT_EQS AND T_INCORRECT_EQS

t_correct_eqs     = NewType('t_correct_eqs',     list[str])
t_incorrect_eqs   = NewType('t_incorrect_eqs',   list[str])



# DATA CLASS C_ELEMENT

@dataclasses.dataclass(frozen=True,kw_only=True)
class c_element:
    ci : F[int] # cycle index
    ri : F[int] # right index
    def __post_init__(self) -> None:
        assert self.ci >= 1, self.ci
        assert self.ri >= 1, self.ri
    def __repr__(self) -> str:
        return f'a[{self.ci},{self.ri}]'



```

```
# DATA CLASS C_CYCLE

@dataclasses.dataclass(frozen=True,kw_only=True)
class c_cycle:
    ci       : F[int]
    length   : F[int]
    elements : F[tuple[c_element,...]] =
        dataclasses.field(init=False)

    def __post_init__(self) -> None:
        assert self.ci     >= 1, self.ci
        assert self.length >= 1, self.length
        elements : F[tuple[c_element,...]] = \
            tuple(c_element(ci=self.ci,ri=ri) for ri in
                range(1,self.length+1))
        object.__setattr__(self,'elements',elements)

    def element(self, ri: int) -> c_element:
        assert ri <= self.length, (ri,self.length)
        return self.elements[ri-1]

    def S(self, a: c_element) -> c_element:
        assert a.ci == self.ci, (a,self.ci)
        assert a.ri <= self.length
        if a.ri == self.length:
            return self.elements[0]
        else:
            return self.elements[a.ri]

    def P(self, a: c_element) -> c_element:
        assert a.ci == self.ci, (a,self.ci)
        assert a.ri <= self.length
        if a.ri == 1:
            return self.elements[-1]
        else:
            return self.elements[a.ri-2]

    def it_S(self, a: c_element, iterations: int) ->
        c_element:
```

```
        if iterations < 0:
            return self.it_P(a,-iterations)
        assert a.ci == self.ci, (a,self.ci)
        assert a.ri <= self.length
        return self.elements[(a.ri-1+iterations) % self.length]

    def it_P(self, a: c_element, iterations: int) -> c_element:
        if iterations < 0:
            return self.it_S(a,-iterations)
        assert a.ci == self.ci, (a, self.ci)
        assert a.ri <= self.length
        return self.elements[(a.ri-1-iterations) % self.length]

    def __repr__(self) -> str:
        return \
            f'A[{self.ci}]' + \
           ' = '         +                               \
            '{' + ', '.join(str(a) for a in self.elements) + '}'



# DATA CLASS CYCLE STRUCTURE

@dataclasses.dataclass(frozen=True,kw_only=True,eq=False)
class c_cycle_structure:
    cycle_lengths : F[tuple[int,...]]
    cycles        : F[tuple[c_cycle,...]]
    elements      : F[tuple[c_element,...]]
    no_of_cycles  : F[int]
    size          : F[int]

    def __init__(self, *, cycle_lengths: tuple[int,...]) -> None:
```

```
        no_of_cycles : F[int]                  =
            len(cycle_lengths)
        cycles       : F[tuple[c_cycle,...]]   = tuple(
            c_cycle(ci=ci,length= ) for (ci, ) in
                enumerate(cycle_lengths,1)
        )
        elements     : F[tuple[c_element,...]] = tuple(
            a for c in cycles for a in c.elements
        )
        assert no_of_cycles >= 1
        assert all( >= 1 for  in cycle_lengths),
            cycle_lengths
        assert \
            all( 1 >= 2 for 1, 2 in
                zip(cycle_lengths,cycle_lengths[1:])),\
            cycle_lengths
        ob-
            ject.__setattr__(self,'cycle_lengths',cycle_lengths)
        object.__setattr__(self,'cycles',       cycles)
        object.__setattr__(self,'elements',     elements)
        object.__setattr__(self,'no_of_cycles',
            no_of_cycles)
        object.__setattr__(self,'size',
            len(self.elements))

    def cycle(self, ci: int) -> c_cycle:
        assert 1 <= ci <= self.no_of_cycles,
            (ci,self.no_of_cycles)
        return self.cycles[ci-1]

    def element(self, *, ci: int, ri: int) -> c_element:
        assert 1 <= ci <= self.no_of_cycles,
            (ci,self.no_of_cycles)
        assert \
            1 <= ri <= self.cycles[ci-1].length,\
            (ci, ri, self.cycles[ci-1].length)
        return self.cycles[ci-1].elements[ri-1]

    def S(self, a: c_element) -> c_element:
```

```
        assert a.ci <= self.no_of_cycles, (a,self.no_of_cycles)
        return self.cycles[a.ci-1].S(a)

    def P(self, a: c_element) -> c_element:
        assert a.ci <= self.no_of_cycles, (a,self.no_of_cycles)
        return self.cycles[a.ci-1].P(a)

    def it_S(self, a: c_element, iterations: int) -> c_element:
        assert a.ci <= self.no_of_cycles, (a,self.no_of_cycles)
        return self.cycles[a.ci-1].it_S(a,iterations)

    def it_P(self, a: c_element, iterations: int) -> c_element:
        assert a.ci <= self.no_of_cycles, (a,self.no_of_cycles)
        return self.cycles[a.ci-1].it_P(a,iterations)

    def __repr__(self) -> str:
        return \
            'A = '                                                +\
            '+'.join(f'A[{ci}]' for ci in (c.ci for c in self.cycles)) +\
            '\n'                                                  +\
            '\n'.join(str(cycle) for cycle in self.cycles)        +\
            '\n'                                                  +\
            '\n'.join(
                '\n'.join(f'S{a} = {self.S(a)}' for a in cycle.elements)
                for cycle in self.cycles
            )

```

```


# TYPE ALIAS T_PLUS

ta_plus  : TypeAlias =
    dict[tuple[c_element,c_element],c_element]



# C_PLUS_REDUCT

@dataclasses.dataclass(frozen=True, kw_only=True)
class c_plus_reduct(c_cycle_structure):

    _plus : F[ta_plus]

    def __init__(self, *, cycle_lengths: tuple[int,...],
        plus: ta_plus) -> None:
        super().__init__(cycle_lengths=cycle_lengths)
        assert \
            set(plus.keys()) == set(product(self.elements,
                repeat=2)),\
            (set(plus.keys()), set(product(self.elements,
                repeat=2)))
        object.__setattr__(self, '_plus', plus)

    @classmethod
    def from_cycle_structure(
        cls,
        cycle_structure : c_cycle_structure,
        plus            : ta_plus,
    ) -> Self:
        return
            cls(cycle_lengths=cycle_structure.cycle_lengths,
            plus=plus)

    def plus(self, a: c_element, b: c_element) ->
        c_element:
```

```
        assert a.ci <= self.no_of_cycles,       (a.ci,
            self.no_of_cycles)
        assert b.ci <= self.no_of_cycles,       (b.ci,
            self.no_of_cycles)
        assert a.ri <= self.cycle(a.ci).length, (a.ri,
            self.cycle(a.ci).length)
        assert b.ri <= self.cycle(b.ci).length, (b.ri,
            self.cycle(b.ci).length)
        return self._plus[(a,b)]

    def it_right_plus(
        self, *, add_to: c_element, add_with: c_element,
            iterations: int,
    ) -> c_element:
        assert iterations >= 0, iterations
        if iterations == 0:
            return add_to
        return self.it_right_plus(
            add_to     = self.plus(add_to,add_with),
            add_with   = add_with,
            iterations = iterations-1,
        )

    def __repr__(self) -> str:
        return \
            super().__repr__() +
                \
            '\n' +
                \
            '\n'.join(
                '\n'.join(
                    f'{a}+{b} = {self.plus(a,b)}'
                    for b,a in product(cycle.elements,
                        self.elements)
                )
                for cycle in self.cycles
            )


```

```

# FUNCTION MODELS_Q5

def models_Q5(pr : c_plus_reduct) -> tuple[t_correct_eqs,t_incorrect_eqs]:
    correct_eqs   : t_correct_eqs   = t_correct_eqs([])
    incorrect_eqs : t_incorrect_eqs = t_incorrect_eqs([])
    for α in pr.elements:
        for β in pr.elements:
            α_Sβ : F[c_element] = pr.plus(α, pr.S(β))
            S_αβ : F[c_element] = pr.S(pr.plus(α,β))
            eq   : str          = f'{α}+S({β}) = {α}+{pr.S(β)} = {α_Sβ}'
            if α_Sβ == S_αβ:
                eq += ' = '
            else:
                eq += f' =/{S_αβ} = '
            eq += f'S({pr.plus(α,β)}) = S({α}+{β})'
            if α_Sβ == S_αβ:
                correct_eqs.append(eq)
            else:
                incorrect_eqs.append(eq)
    return (correct_eqs,incorrect_eqs)



# FUNCTION IS_COMMUTATIVE

def is_commutative(pr: c_plus_reduct) -> tuple[t_correct_eqs,t_incorrect_eqs]:
    correct_eqs   : t_correct_eqs   = t_correct_eqs([])
    incorrect_eqs : t_incorrect_eqs = t_incorrect_eqs([])
    for α in pr.elements:
        for β in pr.elements:
            αβ : F[c_element] = pr.plus(α,β)
            βα : F[c_element] = pr.plus(β,α)
            eq : str = f'{α}+{β} = {αβ}'
            if αβ == βα:
                eq += ' = '
```

```
            else:
                eq += f' =/{βα} = '
            eq += f'{β}+{α}'
            if αβ == βα:
                correct_eqs.append(eq)
            else:
                incorrect_eqs.append(eq)
    return (correct_eqs,incorrect_eqs)



# FUNCTION IS_ASSOCIATIVE

def is_associative(pr: c_plus_reduct) ->
    tuple[t_correct_eqs, t_incorrect_eqs]:
    correct_eqs   : t_correct_eqs   = t_correct_eqs([])
    incorrect_eqs : t_incorrect_eqs = t_incorrect_eqs([])
    for α in pr.elements:
        for β in pr.elements:
            for γ in pr.elements:
                αβ   : F[c_element] = pr.plus(α,β)
                αβ_γ : F[c_element] = pr.plus(αβ,γ)
                βγ   : F[c_element] = pr.plus(β,γ)
                α_βγ : F[c_element] = pr.plus(α,βγ)
                eq : str = f'({α}+{β})+{γ} = {αβ}+{γ} =
                    {αβ_γ}'
                if αβ_γ == α_βγ:
                    eq += ' = '
                else:
                    eq += f' =/{α_βγ} = '
                eq += f'{α}+{βγ} = {α}+({β}+{γ})'
                if αβ_γ == α_βγ:
                    correct_eqs.append(eq)
                else:
                    incorrect_eqs.append(eq)
    return (correct_eqs,incorrect_eqs)
```

```
# FUNCTION IS_EXPANDABLE

def is_expandable(pr: c_plus_reduct) ->
    tuple[t_correct_eqs,t_incorrect_eqs]:
    correct_eqs   : t_correct_eqs   = t_correct_eqs([])
    incorrect_eqs : t_incorrect_eqs = t_incorrect_eqs([])
    for a in pr.elements:
        for ci in range(1,pr.no_of_cycles+1):
            for b in pr.elements:
                a_plus_b_cycle_length_times : F[c_element]
                    = pr.it_right_plus(

                          add_to=a,add_with=b,iterations=pr.cycle(ci).length
                )
                alternative_found : F[bool] = b ==
                    a_plus_b_cycle_length_times
                eq : str = \
                  f'{a}×{pr.element(ci=ci,ri=1)}  {b}'   \
                   +                                     \
                  (' = ' if alternative_found else ' =/') \
                   +                                     \
                   '('*pr.cycle(ci).length               \
                   +                                     \
                    f'{a}+'
                for _ in range(pr.cycle(ci).length-1):
                    eq += f'{b})+'
                eq += f'{b})'
                if alternative_found:
                    correct_eqs.append(eq)
                else:
                    incorrect_eqs.append(eq)
    return (correct_eqs,incorrect_eqs)



# MAIN

## OUR PLUS REDUCTS

```

```
### ELEMENTS USED

a11 : F[c_element] = c_element(ci=1,ri=1)
a12 : F[c_element] = c_element(ci=1,ri=2)
a21 : F[c_element] = c_element(ci=2,ri=1)
a31 : F[c_element] = c_element(ci=3,ri=1)

### PR_NOT_COMMUTATIVE_EXPANDABLE

# this one just to test that function is_expandable works
    as expected
pr_not_commutative_expandable : F[c_plus_reduct] =
    c_plus_reduct(
    cycle_lengths = (2,),
    plus          = {
        (a11,a11): a11,
        (a12,a11): a11,
        (a11,a12): a12,
        (a12,a12): a12,
    }
)

### PR_COMMUTATIVE_ASSOCIATIVE

pr_commutative_associative : F[c_plus_reduct] =
    c_plus_reduct(
    cycle_lengths = (1,1),
    plus          = {
        (a11,a11): a11,
        (a21,a11): a21,
        (a11,a21): a21,
        (a21,a21): a11,
    }
)

### PR_COMMUTATIVE_NOT_ASSOCIATIVE

pr_commutative_not_associative : F[c_plus_reduct] =
    c_plus_reduct(
```

```
    cycle_lengths = (1,1),
    plus          = {
        (a11,a11): a21,
        (a21,a11): a11,
        (a11,a21): a11,
        (a21,a21): a11,
    }
)

### PR_NOT_COMMUTATIVE_ASSOCIATIVE

pr_not_commutative_associative : F[c_plus_reduct] =
    c_plus_reduct(
    cycle_lengths = (2,1,1),
    plus          = {
        (a11,a11): a11,
        (a12,a11): a11,
        (a21,a11): a21,
        (a31,a11): a31,
        (a11,a12): a12,
        (a12,a12): a12,
        (a21,a12): a21,
        (a31,a12): a31,
        (a11,a21): a21,
        (a12,a21): a21,
        (a21,a21): a21,
        (a31,a21): a31,
        (a11,a31): a31,
        (a12,a31): a31,
        (a21,a31): a31,
        (a31,a31): a21,
    }
)

### PR_NOT_COMMUTATIVE_NOT_ASSOCIATIVE

pr_not_commutative_not_associative : F[c_plus_reduct] =
    c_plus_reduct(
    cycle_lengths = (2,),
```

```
    plus          = {
        (a11,a11): a12,
        (a12,a11): a12,
        (a11,a12): a11,
        (a12,a12): a11,
    }
)




## FUNCTION CHECK_PR

def check_pr(pr: c_plus_reduct) -> None:
    print(pr)
    print()
    print('Models (Q5)?',end=' ')
    models_Q5_res           :
        F[tuple[t_correct_eqs,t_incorrect_eqs]] = \
        models_Q5(pr)
    models_Q5_correct_eqs   : F[t_correct_eqs]
        = \
      models_Q5_res[0]
    models_Q5_incorrect_eqs : F[t_incorrect_eqs]
        = \
      models_Q5_res[1]
    if models_Q5_incorrect_eqs != []:
        print('No, a counterexample:')
        print(models_Q5_incorrect_eqs[0])
    else:
        print('Yes:')
        for correct_eq in models_Q5_correct_eqs:
            print(correct_eq)
    print()
    print('Is expandable?',end=' ')
    is_expandable_res           :
        F[tuple[t_correct_eqs,t_incorrect_eqs]] = \
        is_expandable(pr)
```

```
    is_expandable_correct_eqs   : F[t_correct_eqs]
        = \
       is_expandable_res[0]
    is_expandable_incorrect_eqs : F[t_incorrect_eqs]
        = \
       is_expandable_res[1]
    if is_expandable_incorrect_eqs != []:
        print('No:')
        for incorrect_eq in is_expandable_incorrect_eqs:
            print(incorrect_eq)
    else:
        print('Yes, alternatives:')
        for correct_eq in is_expandable_correct_eqs:
            print(correct_eq)
    print()
    print('Is commutative?',end=' ')
    is_commutative_res           :
        F[tuple[t_correct_eqs,t_incorrect_eqs]] = \
        is_commutative(pr)
    is_commutative_correct_eqs   : F[t_correct_eqs]
        = \
       is_commutative_res[0]
    is_commutative_incorrect_eqs : F[t_incorrect_eqs]
        = \
       is_commutative_res[1]
    if is_commutative_incorrect_eqs != []:
        print('No, a counterexample:')
        print(is_commutative_incorrect_eqs[0])
    else:
        print('Yes:')
        for correct_eq in is_commutative_correct_eqs:
            print(correct_eq)
    print()
    print('Is associative?',end=' ')
    is_associative_res           :
        F[tuple[t_correct_eqs,t_incorrect_eqs]] = \
        is_associative(pr)
    is_associative_correct_eqs   : F[t_correct_eqs]
        = \
```

```
        is_associative_res[0]
    is_associative_incorrect_eqs : F[t_incorrect_eqs]
        = \
        is_associative_res[1]
    if is_associative_incorrect_eqs != []:
        print('No, a counterexample:')
        print(is_associative_incorrect_eqs[0])
    else:
        print('Yes:')
        for correct_eq in is_associative_correct_eqs:
            print(correct_eq)


## IF __NAME__ == '__MAIN__':

if __name__ == '__main__':
    print('A non-commutative expandable non-standard
        part:')
    print()
    check_pr(pr_not_commutative_expandable)
    print()
    print('---')
    print()
    print('A commutative associative non-expandable
        non-standard part:')
    print()
    check_pr(pr_commutative_associative)
    print()
    print('---')
    print()
    print('A commutative non-associative non-expandable
        non-standard part:')
    print()
    check_pr(pr_commutative_not_associative)
    print()
    print('---')
    print()
    print('A non-commutative associative non-expandable
        non-standard part:')
```

```
    check_pr(pr_not_commutative_associative)
    print()
    print('---')
    print()
    print('A non-commutative non-associative non-expandable non-standard part:')
    print()
    check_pr(pr_not_commutative_not_associative)
```

§ 5.D.2 Output

Running Python 3.13 with the above source as input produces the following output.

```
A non-commutative expandable non-standard part:

A = A[1]
A[1] = {a[1,1], a[1,2]}
Sa[1,1] = a[1,2]
Sa[1,2] = a[1,1]
a[1,1]+a[1,1] = a[1,1]
a[1,2]+a[1,1] = a[1,1]
a[1,1]+a[1,2] = a[1,2]
a[1,2]+a[1,2] = a[1,2]

Models (Q5)? Yes:
a[1,1]+S(a[1,1]) = a[1,1]+a[1,2] = a[1,2] = S(a[1,1]) = S(a[1,1]+a[1,1])
a[1,1]+S(a[1,2]) = a[1,1]+a[1,1] = a[1,1] = S(a[1,2]) = S(a[1,1]+a[1,2])
a[1,2]+S(a[1,1]) = a[1,2]+a[1,2] = a[1,2] = S(a[1,1]) = S(a[1,2]+a[1,1])
a[1,2]+S(a[1,2]) = a[1,2]+a[1,1] = a[1,1] = S(a[1,2]) = S(a[1,2]+a[1,2])

Is expandable? Yes, alternatives:
a[1,1]×a[1,1]   a[1,1] = ((a[1,1]+a[1,1])+a[1,1])
a[1,1]×a[1,1]   a[1,2] = ((a[1,1]+a[1,2])+a[1,2])
```

```
a[1,2]×a[1,1]   a[1,1] = ((a[1,2]+a[1,1])+a[1,1])
a[1,2]×a[1,1]   a[1,2] = ((a[1,2]+a[1,2])+a[1,2])

Is commutative? No, a counterexample:
a[1,1]+a[1,2] = a[1,2] =/a[1,1] = a[1,2]+a[1,1]

Is associative? Yes:
(a[1,1]+a[1,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[1,1]+a[1,1])
(a[1,1]+a[1,1])+a[1,2] = a[1,1]+a[1,2] = a[1,2] =
    a[1,1]+a[1,2] = a[1,1]+(a[1,1]+a[1,2])
(a[1,1]+a[1,2])+a[1,1] = a[1,2]+a[1,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[1,2]+a[1,1])
(a[1,1]+a[1,2])+a[1,2] = a[1,2]+a[1,2] = a[1,2] =
    a[1,1]+a[1,2] = a[1,1]+(a[1,2]+a[1,2])
(a[1,2]+a[1,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
    a[1,2]+a[1,1] = a[1,2]+(a[1,1]+a[1,1])
(a[1,2]+a[1,1])+a[1,2] = a[1,1]+a[1,2] = a[1,2] =
    a[1,2]+a[1,2] = a[1,2]+(a[1,1]+a[1,2])
(a[1,2]+a[1,2])+a[1,1] = a[1,2]+a[1,1] = a[1,1] =
    a[1,2]+a[1,1] = a[1,2]+(a[1,2]+a[1,1])
(a[1,2]+a[1,2])+a[1,2] = a[1,2]+a[1,2] = a[1,2] =
    a[1,2]+a[1,2] = a[1,2]+(a[1,2]+a[1,2])

---

A commutative associative non-expandable non-standard
    part:

A = A[1]+A[2]
A[1] = {a[1,1]}
A[2] = {a[2,1]}
Sa[1,1] = a[1,1]
Sa[2,1] = a[2,1]
a[1,1]+a[1,1] = a[1,1]
a[2,1]+a[1,1] = a[2,1]
a[1,1]+a[2,1] = a[2,1]
a[2,1]+a[2,1] = a[1,1]
```

```
Models (Q5)? Yes:
a[1,1]+S(a[1,1]) = a[1,1]+a[1,1] = a[1,1] = S(a[1,1]) =
    S(a[1,1]+a[1,1])
a[1,1]+S(a[2,1]) = a[1,1]+a[2,1] = a[2,1] = S(a[2,1]) =
    S(a[1,1]+a[2,1])
a[2,1]+S(a[1,1]) = a[2,1]+a[1,1] = a[2,1] = S(a[2,1]) =
    S(a[2,1]+a[1,1])
a[2,1]+S(a[2,1]) = a[2,1]+a[2,1] = a[1,1] = S(a[1,1]) =
    S(a[2,1]+a[2,1])

Is expandable? No:
a[2,1]×a[1,1]   a[1,1] =/(a[2,1]+a[1,1])
a[2,1]×a[1,1]   a[2,1] =/(a[2,1]+a[2,1])
a[2,1]×a[2,1]   a[1,1] =/(a[2,1]+a[1,1])
a[2,1]×a[2,1]   a[2,1] =/(a[2,1]+a[2,1])

Is commutative? Yes:
a[1,1]+a[1,1] = a[1,1] = a[1,1]+a[1,1]
a[1,1]+a[2,1] = a[2,1] = a[2,1]+a[1,1]
a[2,1]+a[1,1] = a[2,1] = a[1,1]+a[2,1]
a[2,1]+a[2,1] = a[1,1] = a[2,1]+a[2,1]

Is associative? Yes:
(a[1,1]+a[1,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[1,1]+a[1,1])
(a[1,1]+a[1,1])+a[2,1] = a[1,1]+a[2,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[1,1]+a[2,1])
(a[1,1]+a[2,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[2,1]+a[1,1])
(a[1,1]+a[2,1])+a[2,1] = a[2,1]+a[2,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[2,1]+a[2,1])
(a[2,1]+a[1,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[2,1]+a[1,1] = a[2,1]+(a[1,1]+a[1,1])
(a[2,1]+a[1,1])+a[2,1] = a[2,1]+a[2,1] = a[1,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[1,1]+a[2,1])
(a[2,1]+a[2,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[2,1]+a[1,1])
(a[2,1]+a[2,1])+a[2,1] = a[1,1]+a[2,1] = a[2,1] =
    a[2,1]+a[1,1] = a[2,1]+(a[2,1]+a[2,1])
```

```

---

A commutative non-associative non-expandable non-standard
    part:

A = A[1]+A[2]
A[1] = {a[1,1]}
A[2] = {a[2,1]}
Sa[1,1] = a[1,1]
Sa[2,1] = a[2,1]
a[1,1]+a[1,1] = a[2,1]
a[2,1]+a[1,1] = a[1,1]
a[1,1]+a[2,1] = a[1,1]
a[2,1]+a[2,1] = a[1,1]

Models (Q5)? Yes:
a[1,1]+S(a[1,1]) = a[1,1]+a[1,1] = a[2,1] = S(a[2,1]) =
    S(a[1,1]+a[1,1])
a[1,1]+S(a[2,1]) = a[1,1]+a[2,1] = a[1,1] = S(a[1,1]) =
    S(a[1,1]+a[2,1])
a[2,1]+S(a[1,1]) = a[2,1]+a[1,1] = a[1,1] = S(a[1,1]) =
    S(a[2,1]+a[1,1])
a[2,1]+S(a[2,1]) = a[2,1]+a[2,1] = a[1,1] = S(a[1,1]) =
    S(a[2,1]+a[2,1])

Is expandable? No:
a[1,1]×a[1,1]   a[1,1] =/(a[1,1]+a[1,1])
a[1,1]×a[1,1]   a[2,1] =/(a[1,1]+a[2,1])
a[1,1]×a[2,1]   a[1,1] =/(a[1,1]+a[1,1])
a[1,1]×a[2,1]   a[2,1] =/(a[1,1]+a[2,1])
a[2,1]×a[1,1]   a[2,1] =/(a[2,1]+a[2,1])
a[2,1]×a[2,1]   a[2,1] =/(a[2,1]+a[2,1])

Is commutative? Yes:
a[1,1]+a[1,1] = a[2,1] = a[1,1]+a[1,1]
a[1,1]+a[2,1] = a[1,1] = a[2,1]+a[1,1]
a[2,1]+a[1,1] = a[1,1] = a[1,1]+a[2,1]
a[2,1]+a[2,1] = a[1,1] = a[2,1]+a[2,1]
```

```

Is associative? No, a counterexample:
(a[1,1]+a[1,1])+a[2,1] = a[2,1]+a[2,1] = a[1,1] =/a[2,1] = a[1,1]+a[1,1] = a[1,1]+(a[1,1]+a[2,1])

---

A non-commutative associative non-expandable non-standard part:
A = A[1]+A[2]+A[3]
A[1] = {a[1,1], a[1,2]}
A[2] = {a[2,1]}
A[3] = {a[3,1]}
Sa[1,1] = a[1,2]
Sa[1,2] = a[1,1]
Sa[2,1] = a[2,1]
Sa[3,1] = a[3,1]
a[1,1]+a[1,1] = a[1,1]
a[1,2]+a[1,1] = a[1,1]
a[2,1]+a[1,1] = a[2,1]
a[3,1]+a[1,1] = a[3,1]
a[1,1]+a[1,2] = a[1,2]
a[1,2]+a[1,2] = a[1,2]
a[2,1]+a[1,2] = a[2,1]
a[3,1]+a[1,2] = a[3,1]
a[1,1]+a[2,1] = a[2,1]
a[1,2]+a[2,1] = a[2,1]
a[2,1]+a[2,1] = a[2,1]
a[3,1]+a[2,1] = a[3,1]
a[1,1]+a[3,1] = a[3,1]
a[1,2]+a[3,1] = a[3,1]
a[2,1]+a[3,1] = a[3,1]
a[3,1]+a[3,1] = a[2,1]

Models (Q5)? Yes:
a[1,1]+S(a[1,1]) = a[1,1]+a[1,2] = a[1,2] = S(a[1,1]) = S(a[1,1]+a[1,1])
a[1,1]+S(a[1,2]) = a[1,1]+a[1,1] = a[1,1] = S(a[1,2]) = S(a[1,1]+a[1,2])
```

```
a[1,1]+S(a[2,1]) = a[1,1]+a[2,1] = a[2,1] = S(a[2,1]) =
    S(a[1,1]+a[2,1])
a[1,1]+S(a[3,1]) = a[1,1]+a[3,1] = a[3,1] = S(a[3,1]) =
    S(a[1,1]+a[3,1])
a[1,2]+S(a[1,1]) = a[1,2]+a[1,2] = a[1,2] = S(a[1,1]) =
    S(a[1,2]+a[1,1])
a[1,2]+S(a[1,2]) = a[1,2]+a[1,1] = a[1,1] = S(a[1,2]) =
    S(a[1,2]+a[1,2])
a[1,2]+S(a[2,1]) = a[1,2]+a[2,1] = a[2,1] = S(a[2,1]) =
    S(a[1,2]+a[2,1])
a[1,2]+S(a[3,1]) = a[1,2]+a[3,1] = a[3,1] = S(a[3,1]) =
    S(a[1,2]+a[3,1])
a[2,1]+S(a[1,1]) = a[2,1]+a[1,2] = a[2,1] = S(a[2,1]) =
    S(a[2,1]+a[1,1])
a[2,1]+S(a[1,2]) = a[2,1]+a[1,1] = a[2,1] = S(a[2,1]) =
    S(a[2,1]+a[1,2])
a[2,1]+S(a[2,1]) = a[2,1]+a[2,1] = a[2,1] = S(a[2,1]) =
    S(a[2,1]+a[2,1])
a[2,1]+S(a[3,1]) = a[2,1]+a[3,1] = a[3,1] = S(a[3,1]) =
    S(a[2,1]+a[3,1])
a[3,1]+S(a[1,1]) = a[3,1]+a[1,2] = a[3,1] = S(a[3,1]) =
    S(a[3,1]+a[1,1])
a[3,1]+S(a[1,2]) = a[3,1]+a[1,1] = a[3,1] = S(a[3,1]) =
    S(a[3,1]+a[1,2])
a[3,1]+S(a[2,1]) = a[3,1]+a[2,1] = a[3,1] = S(a[3,1]) =
    S(a[3,1]+a[2,1])
a[3,1]+S(a[3,1]) = a[3,1]+a[3,1] = a[2,1] = S(a[2,1]) =
    S(a[3,1]+a[3,1])

Is expandable? No:
a[1,1]×a[1,1]   a[3,1] =/((a[1,1]+a[3,1])+a[3,1])
a[1,2]×a[1,1]   a[3,1] =/((a[1,2]+a[3,1])+a[3,1])
a[2,1]×a[1,1]   a[1,1] =/((a[2,1]+a[1,1])+a[1,1])
a[2,1]×a[1,1]   a[1,2] =/((a[2,1]+a[1,2])+a[1,2])
a[2,1]×a[1,1]   a[3,1] =/((a[2,1]+a[3,1])+a[3,1])
a[2,1]×a[2,1]   a[1,1] =/(a[2,1]+a[1,1])
a[2,1]×a[2,1]   a[1,2] =/(a[2,1]+a[1,2])
a[2,1]×a[3,1]   a[1,1] =/(a[2,1]+a[1,1])
a[2,1]×a[3,1]   a[1,2] =/(a[2,1]+a[1,2])
```

```
a[3,1]×a[1,1]   a[1,1] =/((a[3,1]+a[1,1])+a[1,1])
a[3,1]×a[1,1]   a[1,2] =/((a[3,1]+a[1,2])+a[1,2])
a[3,1]×a[1,1]   a[2,1] =/((a[3,1]+a[2,1])+a[2,1])
a[3,1]×a[2,1]   a[1,1] =/(a[3,1]+a[1,1])
a[3,1]×a[2,1]   a[1,2] =/(a[3,1]+a[1,2])
a[3,1]×a[2,1]   a[2,1] =/(a[3,1]+a[2,1])
a[3,1]×a[2,1]   a[3,1] =/(a[3,1]+a[3,1])
a[3,1]×a[3,1]   a[1,1] =/(a[3,1]+a[1,1])
a[3,1]×a[3,1]   a[1,2] =/(a[3,1]+a[1,2])
a[3,1]×a[3,1]   a[2,1] =/(a[3,1]+a[2,1])
a[3,1]×a[3,1]   a[3,1] =/(a[3,1]+a[3,1])

Is commutative? No, a counterexample:
a[1,1]+a[1,2] = a[1,2] =/a[1,1] = a[1,2]+a[1,1]

Is associative? Yes:
(a[1,1]+a[1,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[1,1]+a[1,1])
(a[1,1]+a[1,1])+a[1,2] = a[1,1]+a[1,2] = a[1,2] =
    a[1,1]+a[1,2] = a[1,1]+(a[1,1]+a[1,2])
(a[1,1]+a[1,1])+a[2,1] = a[1,1]+a[2,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[1,1]+a[2,1])
(a[1,1]+a[1,1])+a[3,1] = a[1,1]+a[3,1] = a[3,1] =
    a[1,1]+a[3,1] = a[1,1]+(a[1,1]+a[3,1])
(a[1,1]+a[1,2])+a[1,1] = a[1,2]+a[1,1] = a[1,1] =
    a[1,1]+a[1,1] = a[1,1]+(a[1,2]+a[1,1])
(a[1,1]+a[1,2])+a[1,2] = a[1,2]+a[1,2] = a[1,2] =
    a[1,1]+a[1,2] = a[1,1]+(a[1,2]+a[1,2])
(a[1,1]+a[1,2])+a[2,1] = a[1,2]+a[2,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[1,2]+a[2,1])
(a[1,1]+a[1,2])+a[3,1] = a[1,2]+a[3,1] = a[3,1] =
    a[1,1]+a[3,1] = a[1,1]+(a[1,2]+a[3,1])
(a[1,1]+a[2,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[2,1]+a[1,1])
(a[1,1]+a[2,1])+a[1,2] = a[2,1]+a[1,2] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[2,1]+a[1,2])
(a[1,1]+a[2,1])+a[2,1] = a[2,1]+a[2,1] = a[2,1] =
    a[1,1]+a[2,1] = a[1,1]+(a[2,1]+a[2,1])
```

```
199 (a[1,1]+a[2,1])+a[3,1] = a[2,1]+a[3,1] = a[3,1] =
        a[1,1]+a[3,1] = a[1,1]+(a[2,1]+a[3,1])
200 (a[1,1]+a[3,1])+a[1,1] = a[3,1]+a[1,1] = a[3,1] =
        a[1,1]+a[3,1] = a[1,1]+(a[3,1]+a[1,1])
201 (a[1,1]+a[3,1])+a[1,2] = a[3,1]+a[1,2] = a[3,1] =
        a[1,1]+a[3,1] = a[1,1]+(a[3,1]+a[1,2])
202 (a[1,1]+a[3,1])+a[2,1] = a[3,1]+a[2,1] = a[3,1] =
        a[1,1]+a[3,1] = a[1,1]+(a[3,1]+a[2,1])
203 (a[1,1]+a[3,1])+a[3,1] = a[3,1]+a[3,1] = a[2,1] =
        a[1,1]+a[2,1] = a[1,1]+(a[3,1]+a[3,1])
204 (a[1,2]+a[1,1])+a[1,1] = a[1,1]+a[1,1] = a[1,1] =
        a[1,2]+a[1,1] = a[1,2]+(a[1,1]+a[1,1])
205 (a[1,2]+a[1,1])+a[1,2] = a[1,1]+a[1,2] = a[1,2] =
        a[1,2]+a[1,2] = a[1,2]+(a[1,1]+a[1,2])
206 (a[1,2]+a[1,1])+a[2,1] = a[1,1]+a[2,1] = a[2,1] =
        a[1,2]+a[2,1] = a[1,2]+(a[1,1]+a[2,1])
207 (a[1,2]+a[1,1])+a[3,1] = a[1,1]+a[3,1] = a[3,1] =
        a[1,2]+a[3,1] = a[1,2]+(a[1,1]+a[3,1])
208 (a[1,2]+a[1,2])+a[1,1] = a[1,2]+a[1,1] = a[1,1] =
        a[1,2]+a[1,1] = a[1,2]+(a[1,2]+a[1,1])
209 (a[1,2]+a[1,2])+a[1,2] = a[1,2]+a[1,2] = a[1,2] =
        a[1,2]+a[1,2] = a[1,2]+(a[1,2]+a[1,2])
210 (a[1,2]+a[1,2])+a[2,1] = a[1,2]+a[2,1] = a[2,1] =
        a[1,2]+a[2,1] = a[1,2]+(a[1,2]+a[2,1])
211 (a[1,2]+a[1,2])+a[3,1] = a[1,2]+a[3,1] = a[3,1] =
        a[1,2]+a[3,1] = a[1,2]+(a[1,2]+a[3,1])
212 (a[1,2]+a[2,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
        a[1,2]+a[2,1] = a[1,2]+(a[2,1]+a[1,1])
213 (a[1,2]+a[2,1])+a[1,2] = a[2,1]+a[1,2] = a[2,1] =
        a[1,2]+a[2,1] = a[1,2]+(a[2,1]+a[1,2])
214 (a[1,2]+a[2,1])+a[2,1] = a[2,1]+a[2,1] = a[2,1] =
        a[1,2]+a[2,1] = a[1,2]+(a[2,1]+a[2,1])
215 (a[1,2]+a[2,1])+a[3,1] = a[2,1]+a[3,1] = a[3,1] =
        a[1,2]+a[3,1] = a[1,2]+(a[2,1]+a[3,1])
216 (a[1,2]+a[3,1])+a[1,1] = a[3,1]+a[1,1] = a[3,1] =
        a[1,2]+a[3,1] = a[1,2]+(a[3,1]+a[1,1])
217 (a[1,2]+a[3,1])+a[1,2] = a[3,1]+a[1,2] = a[3,1] =
        a[1,2]+a[3,1] = a[1,2]+(a[3,1]+a[1,2])
```

```
(a[1,2]+a[3,1])+a[2,1] = a[3,1]+a[2,1] = a[3,1] =
    a[1,2]+a[3,1] = a[1,2]+(a[3,1]+a[2,1])
(a[1,2]+a[3,1])+a[3,1] = a[3,1]+a[3,1] = a[2,1] =
    a[1,2]+a[2,1] = a[1,2]+(a[3,1]+a[3,1])
(a[2,1]+a[1,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[2,1]+a[1,1] = a[2,1]+(a[1,1]+a[1,1])
(a[2,1]+a[1,1])+a[1,2] = a[2,1]+a[1,2] = a[2,1] =
    a[2,1]+a[1,2] = a[2,1]+(a[1,1]+a[1,2])
(a[2,1]+a[1,1])+a[2,1] = a[2,1]+a[2,1] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[1,1]+a[2,1])
(a[2,1]+a[1,1])+a[3,1] = a[2,1]+a[3,1] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[1,1]+a[3,1])
(a[2,1]+a[1,2])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[2,1]+a[1,1] = a[2,1]+(a[1,2]+a[1,1])
(a[2,1]+a[1,2])+a[1,2] = a[2,1]+a[1,2] = a[2,1] =
    a[2,1]+a[1,2] = a[2,1]+(a[1,2]+a[1,2])
(a[2,1]+a[1,2])+a[2,1] = a[2,1]+a[2,1] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[1,2]+a[2,1])
(a[2,1]+a[1,2])+a[3,1] = a[2,1]+a[3,1] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[1,2]+a[3,1])
(a[2,1]+a[2,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[2,1]+a[1,1])
(a[2,1]+a[2,1])+a[1,2] = a[2,1]+a[1,2] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[2,1]+a[1,2])
(a[2,1]+a[2,1])+a[2,1] = a[2,1]+a[2,1] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[2,1]+a[2,1])
(a[2,1]+a[2,1])+a[3,1] = a[2,1]+a[3,1] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[2,1]+a[3,1])
(a[2,1]+a[3,1])+a[1,1] = a[3,1]+a[1,1] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[3,1]+a[1,1])
(a[2,1]+a[3,1])+a[1,2] = a[3,1]+a[1,2] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[3,1]+a[1,2])
(a[2,1]+a[3,1])+a[2,1] = a[3,1]+a[2,1] = a[3,1] =
    a[2,1]+a[3,1] = a[2,1]+(a[3,1]+a[2,1])
(a[2,1]+a[3,1])+a[3,1] = a[3,1]+a[3,1] = a[2,1] =
    a[2,1]+a[2,1] = a[2,1]+(a[3,1]+a[3,1])
(a[3,1]+a[1,1])+a[1,1] = a[3,1]+a[1,1] = a[3,1] =
    a[3,1]+a[1,1] = a[3,1]+(a[1,1]+a[1,1])
```

```
(a[3,1]+a[1,1])+a[1,2] = a[3,1]+a[1,2] = a[3,1] = a[3,1]+a[1,2] = a[3,1]+(a[1,1]+a[1,2])
(a[3,1]+a[1,1])+a[2,1] = a[3,1]+a[2,1] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[1,1]+a[2,1])
(a[3,1]+a[1,1])+a[3,1] = a[3,1]+a[3,1] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[1,1]+a[3,1])
(a[3,1]+a[1,2])+a[1,1] = a[3,1]+a[1,1] = a[3,1] = a[3,1]+a[1,1] = a[3,1]+(a[1,2]+a[1,1])
(a[3,1]+a[1,2])+a[1,2] = a[3,1]+a[1,2] = a[3,1] = a[3,1]+a[1,2] = a[3,1]+(a[1,2]+a[1,2])
(a[3,1]+a[1,2])+a[2,1] = a[3,1]+a[2,1] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[1,2]+a[2,1])
(a[3,1]+a[1,2])+a[3,1] = a[3,1]+a[3,1] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[1,2]+a[3,1])
(a[3,1]+a[2,1])+a[1,1] = a[3,1]+a[1,1] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[2,1]+a[1,1])
(a[3,1]+a[2,1])+a[1,2] = a[3,1]+a[1,2] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[2,1]+a[1,2])
(a[3,1]+a[2,1])+a[2,1] = a[3,1]+a[2,1] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[2,1]+a[2,1])
(a[3,1]+a[2,1])+a[3,1] = a[3,1]+a[3,1] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[2,1]+a[3,1])
(a[3,1]+a[3,1])+a[1,1] = a[2,1]+a[1,1] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[3,1]+a[1,1])
(a[3,1]+a[3,1])+a[1,2] = a[2,1]+a[1,2] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[3,1]+a[1,2])
(a[3,1]+a[3,1])+a[2,1] = a[2,1]+a[2,1] = a[2,1] = a[3,1]+a[3,1] = a[3,1]+(a[3,1]+a[2,1])
(a[3,1]+a[3,1])+a[3,1] = a[2,1]+a[3,1] = a[3,1] = a[3,1]+a[2,1] = a[3,1]+(a[3,1]+a[3,1])

---

A non-commutative non-associative non-expandable non-standard part:

A = A[1]
A[1] = {a[1,1], a[1,2]}
Sa[1,1] = a[1,2]
```

```
Sa[1,2] = a[1,1]
a[1,1]+a[1,1] = a[1,2]
a[1,2]+a[1,1] = a[1,2]
a[1,1]+a[1,2] = a[1,1]
a[1,2]+a[1,2] = a[1,1]

Models (Q5)? Yes:
a[1,1]+S(a[1,1]) = a[1,1]+a[1,2] = a[1,1] = S(a[1,2]) =
    S(a[1,1]+a[1,1])
a[1,1]+S(a[1,2]) = a[1,1]+a[1,1] = a[1,2] = S(a[1,1]) =
    S(a[1,1]+a[1,2])
a[1,2]+S(a[1,1]) = a[1,2]+a[1,2] = a[1,1] = S(a[1,2]) =
    S(a[1,2]+a[1,1])
a[1,2]+S(a[1,2]) = a[1,2]+a[1,1] = a[1,2] = S(a[1,1]) =
    S(a[1,2]+a[1,2])

Is expandable? No:
a[1,1]×a[1,1]   a[1,1] =/((a[1,1]+a[1,1])+a[1,1])
a[1,1]×a[1,1]   a[1,2] =/((a[1,1]+a[1,2])+a[1,2])
a[1,2]×a[1,1]   a[1,1] =/((a[1,2]+a[1,1])+a[1,1])
a[1,2]×a[1,1]   a[1,2] =/((a[1,2]+a[1,2])+a[1,2])

Is commutative? No, a counterexample:
a[1,1]+a[1,2] = a[1,1] =/a[1,2] = a[1,2]+a[1,1]

Is associative? No, a counterexample:
(a[1,1]+a[1,1])+a[1,1] = a[1,2]+a[1,1] = a[1,2] =/a[1,1] =
    a[1,1]+a[1,2] = a[1,1]+(a[1,1]+a[1,1])
```

Bibliography

Bentkamp, Alexander, and Jon Eugster (2025): Natural Number Game.
URL (accessed April 16, 2025):
https://adam.math.hhu.de/#/g/leanprover-community/nng4
Source code URL (accessed April 16, 2025):
https://github.com/leanprover-community/lean4game

Boolos, George S., and Richard C. Jeffrey (1980): Computability and Logic, edition 2; Cambridge University Press; Cambridge.

Brox, Jose (2010): How to locate the paper that established Robinson Arithmetic?. At: MathOverflow; version: September 9, 2023.
URL (accessed 2025-04-20).
https://mathoverflow.net/q/30646.

Buzzard, Kevin (2021): Re: 870: Structural Proof Theory/2. In: The Foundations of Mathematics mailing list.
URL:
https://cs.nyu.edu/pipermail/fom/2021-March/022579.html

Carneiro, Mario (2021): Re: 870: Structural Proof Theory/2. In: The Foundations of Mathematics mailing list.
URL:
https://cs.nyu.edu/pipermail/fom/2021-March/022582.html

Euler, Leonhard (1740): De summis serierum reciprocarum. In: Commentarii academiae scientiarum Petropolitanae, volume 7, pages 123–134.

Friedman, Harvey (2021): 870: Structural Proof Theory/2. In: The Foundations of Mathematics mailing list.
URL:
https://cs.nyu.edu/pipermail/fom/2021-February/022512.html

Gotti, Felix (2020): Irreducibility and factorizations in monoid rings. In: Numerical Semigroups, pages 129–139; edited by Valentina Barucci, Scott Chapman, Marco D'Anna and Ralf Fröberg; Springer International Publishing.

Hetzl, Stefan, and Tin Lok Wong (2018): Some observations on the logical foundations of inductive theorem proving. In: Logical Methods in Computer Science, volume 13, number 4, pages 1–26. (Corrected version of paper originally published November 16, 2017.)

Kaye, Richard (1991): Models of Peano Arithmetic; Clarendon Press; Oxford.

The Lean Focused Research Organization (2025): Lean 4.
URL (accessed April 16, 2025):
https://lean-lang.org/

Quine, W.V (1968): Ontological relativity. In: The Journal of Philosophy, volume 65, number 7, pages 185–212.

Robinson, Raphael M. (1950): An essentially undecidable axiom system. In: Proceedings of the International Congress of Mathematicians, volume I, pages 729–730; American Mathematical Society; Providence 1952.

Tarski, Alfred, and Andrzej Mostowski, and Raphael M. Robinson (1953): Undecidable Theories; North-Holland Publishing Company; Amsterdam.

Westerståhl, Dag (2023): Foundations of Logic: Completeness, Incompleteness, Computability; CSLI Publications; Stanford.