computation of the automaton $\mathcal{A}$, so that:

> a computation is possible in $\mathcal{A}$ iff it can be proved in $\mathcal{L}(\mathcal{A})$

$\mathcal{L}(\mathcal{A})$ has three simple inference rules, where an inference rule will always have one of two forms:

1. $$\frac{q_1 \xrightarrow{\alpha_1} q_1' \quad \dots \quad q_n \xrightarrow{\alpha_n} q_n'}{q \xrightarrow{\alpha} q'}$$

meaning: if we can prove $q_i \xrightarrow{\alpha_i} q_i'$, $1 \le i \le n$, then we can also prove $q \xrightarrow{\alpha} q'$

2. $$\frac{}{q \xrightarrow{\alpha} q'}$$

meaning: $q \xrightarrow{\alpha} q'$ is always true under no extra assumption or, equivalently, $q \xrightarrow{\alpha} q'$ is an axiom.

$\mathcal{L}(\mathcal{A})$ has three inference rules:

1. Action. For each $(q, l, q') \in \xrightarrow{}_{\mathcal{A}}$: $$\frac{}{q \xrightarrow{l} q'}$$

2. Idle. For each $q \in Q$ $$\frac{}{q \xrightarrow{q} q}$$

Transitivity
$$\frac{q \xrightarrow{\alpha} q' \qquad q' \xrightarrow{\beta} q''}{q \xrightarrow{\alpha;\beta} q''}$$

Def: A proof tree in $\mathcal{L}(\mathcal{A})$ is either an axiom $\overline{q \xrightarrow{1} q}$

or $\overline{q \xrightarrow{a} q}$, or a tree of the form:

$$\frac{\begin{array}{cc} T_1 & T_2 \\ q \xrightarrow{\alpha} q' & q' \xrightarrow{\beta} q'' \end{array}}{q \xrightarrow{\alpha;\beta} q''}$$

where the leaves in the proof subtrees T_1 and T_2 are all axioms in $\mathcal{L}(\mathcal{A})$.

Examples: For the automaton $\mathcal{A}$ in pg. 4, we can build the following two proof trees, among many others:

$$\dfrac{\dfrac{\dfrac{A}{q_1 \xrightarrow{a} q_2} \qquad \dfrac{I}{q_2 \xrightarrow{c} q_2}}{q_1 \xrightarrow{a;c} q_2} \qquad \dfrac{I}{q_2 \xrightarrow{c} q_2}}{q_1 \xrightarrow{(a;c);c} q_2}\ T$$

$$\dfrac{\dfrac{\dfrac{A}{q_1 \xrightarrow{a} q_2} \qquad \dfrac{A}{q_2 \xrightarrow{c} q_2}}{q_1 \xrightarrow{a;c} q_2}\ T \qquad \dfrac{\dfrac{A}{q_2 \xrightarrow{c} q_2} \qquad \dfrac{A}{q_2 \xrightarrow{b} q_1}}{q_2 \xrightarrow{c;b} q_1}\ T}{q_1 \xrightarrow{(a;c);(c;b)} q_1}$$

Intuitively, what the computation specification $q_1 \xrightarrow{(a;c);(c;b)} q_1$ specifies is that in $\mathcal{A}$ we can first compute $q_1 \xrightarrow{(a;c)} q_2$ by first performing a and then c, and then compute $q_2 \xrightarrow{c;b} q_1$ by first performing c and then performing b.

Likewise the computation specification $q_1 \xrightarrow{(a;q_2);c} q_2$ specifies that in $\mathcal{A}$ we can first compute $q_1 \xrightarrow{a;q_2} q_2$ by first performing a and then idling in q_2, and then we can perform $q_2 \xrightarrow{c} q_2$.

These computation specifications are obviously very detailed. So, a natural question to ask is:

> When do two computation specification describe the same computation of the automaton $\mathcal{A}$?

For example, it seems intuitive obvious that:

$$q_1 \xrightarrow{(a;c);(c;b)} q_1 \quad q_1 \xrightarrow{a;(c;(c;b))} q_1 \quad q_1 \xrightarrow{((a;c);c);b} q_1$$

and likewise $q_1 \xrightarrow{(a;q_2);c} q_2 \quad q_1 \xrightarrow{a;c} q_2 \quad q_1 \xrightarrow{(q_1;a);c} q_2$

describe in both the cases the same computations of the automaton $\mathcal{A}$ in fig. 4.

Q: Can we capture this notion by defining a computation of an automaton $\mathcal{A}$ an equivalence class $[q \xrightarrow{\alpha} q']_{\equiv}$ of specification specifications under a suitable equivalence relation $\equiv$?

A: Yes! It is the smallest equivalence relation $\equiv$ generated by reflexivity, symmetry and transitivity from the following basic equivalences:

1. Associativity

$$q \xrightarrow{(\alpha;\beta);\gamma} q' \equiv q \xrightarrow{\alpha;(\beta;\gamma)} q'$$

2. Identity

$$q \xrightarrow{\alpha;q'} q' \equiv q \xrightarrow{\alpha} q' \equiv q \xrightarrow{q;\alpha} q'$$

Then we call a computation of $\mathcal{A}$ an equivalence class $[q \xrightarrow{\alpha} q']_{\equiv}$ of computation specifications under the above equivalence relation.

Note that by: (i) dropping parenteses and (ii) dropping idle computations $q \xrightarrow{q} q$ any computation $[q \xrightarrow{\alpha} q']_{\equiv}$ has a very simple representation as either:

1. a basic transition $q \xrightarrow{l} q'$ or
2. an idle computation $q \xrightarrow{q} q$ or
3. a path $q \xrightarrow{l_1;l_2;\cdots;l_n} q'$ with $n \geq 2$.

Therefore, the computations of $\mathcal{A}$ are either (i) idle computations at a state q, $q \xrightarrow{q} q$ or (ii) paths of length $n \geq 1$ in the labeled directed graph $\mathcal{A}$.

Example $[q_1 \xrightarrow{(a;c);(c;b)} q_1]_{\equiv}$ is denoted $q_1 \xrightarrow{a;c;c;b} q_1$

$[q_1 \xrightarrow{(a;q_2);c} q_2]_{\equiv}$ is denoted $q_1 \xrightarrow{a;c} q_2$

Definition Given an automaton $\mathcal{A} = (Q, L, \rightarrow_{\mathcal{A}})$, its:

1. Its labeled directed graph of computation specifications $Spec(\mathcal{A}) = (Q, ProofTerm(\mathcal{A}), \xrightarrow[Spec(\mathcal{A})]{})$ has nodes Q, and directed edges $q \xrightarrow{\alpha} q'$ exactly the computation specifications.
2. Its category of computations $\mathcal{T}_{\mathcal{A}} = (Q, ProofTerm(\mathcal{A})/_{\equiv}, \xrightarrow[\mathcal{T}(\mathcal{A})]{})$ has nodes Q and arrows its computations $[q \xrightarrow{\alpha} q']_{\equiv}$.

We can also write: $Path(\mathcal{A}) = \mathcal{T}_{\mathcal{A}}$, the path category of $\mathcal{A}$.

Q: Why is $\mathcal{T}_A$ called a category, and not just a directed graph?

A: Because it is a directed graph with two additional operations obeying the algebraic laws that define a category:

Definition A (small) category $\mathcal{C} = (Q, L, \to_{\mathcal{C}})$ is a labeled directed graph together with two additional operations:

1. **Identities**: For each $q \in Q$ there is a chosen labeled edge denoted $q \xrightarrow{id_q} q$ called the identity on q.
2. **Composition** For any two labeled edges $q \xrightarrow{\alpha} q'$ and $q' \xrightarrow{\beta} q''$ there is a chosen labeled edge denoted $q \xrightarrow{\alpha;\beta} q''$ denoted their composition

Furthermore, $\mathcal{C}$ satisfies the following equational laws for any $q_1 \xrightarrow{\alpha} q_2$, $q_2 \xrightarrow{\beta} q_3$, $q_3 \xrightarrow{\gamma} q_4$ in $\mathcal{C}$:

Associativity $q_1 \xrightarrow{(\alpha;\beta);\gamma} q_4 = q_1 \xrightarrow{\alpha;(\beta;\gamma)} q_4$

Identity $q_1 \xrightarrow{q_1;\alpha} q_2 = q_1 \xrightarrow{\alpha} q_2 = q_1 \xrightarrow{\alpha;q_2} q_2$

Corollary $\mathcal{T}_A$ is a category

Proof. This immediately follows from the equivalence relation $\equiv$ in pg. 8.

Q: Why is a category $\mathcal{C} = (Q, L, \to_{\mathcal{C}})$ called small?

A: Because there are also, so called large categories, whose both their nodes (called the objects) and their labeled edges $q \xrightarrow{\alpha} q'$ called the arrows of the category are not sets, but classes of sets. For example the classes:

1. $\mathcal{V}$ of all sets
2. Fun of all functions
3. Rel of all relations

are not sets themselves, but only classes in Set Theory. This distinction is needed because of Russel's paradox: assuming that the "set of all sets" exists leads to an easy contradiction.

Definition A large category is a triple $\mathcal{C} = (\mathcal{U}, W, \to_{\mathcal{C}})$ is just a large labeled directed graph whose class of objects in $\mathcal{U}$, and class of arrows is W such that its directed arrows $A \xrightarrow{\alpha} B$ with $A, B \in \mathcal{U}$, $\alpha \in W$ have identity $id_A : A \to A$ and composition $A \xrightarrow{\alpha;\beta} C$ for any $A \xrightarrow{\alpha} B$ and $B \xrightarrow{\beta} C$ satisfying the

Associativity and Identity equational laws. Furthermore, $\mathcal{C}$ is locally small, in the sense that for any $A, B \in \mathcal{U}$ if we define:

$$\mathcal{C}(A,B) = \{ A \xrightarrow{\alpha} B \mid (A, \alpha, B) \in R \}$$

then $\mathcal{C}(A,B)$ is a set.

Examples. 1. The category of sets $\underline{Set} = (\mathcal{V}, Fun, \overrightarrow{Set})$ has as class of objects the class $\mathcal{V}$ of all sets, as its class of labels the class Fun of all functions, and

$$\underline{Set}(A,B) = \{ A \xrightarrow{f} B \mid A, B \in \mathcal{V}, f \in Fun, f \text{ a function } f: A \to B \}$$

we abbreviate $\underline{Set}(A,B)$ to $[A \to B] \in \mathcal{V}$ and call it the function set of functions from A to B

The identity id_A for each $A \in \mathcal{V}$ is the identity function $id_A : A \ni a \mapsto a \in A$. And given $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ then $A \xrightarrow{f;g} C$ is sometimes written $A \xrightarrow{g \circ f} C$ and is the function $f;g : A \ni a \longmapsto g(f(a)) \in C$. The Associativity and Identity axioms are trivial properties of function composition.

2. The category of sets and relations $\underline{Rel} = (\mathcal{V}, Rel, \overrightarrow{Rel})$ has class of objects the class $\mathcal{V}$ of all sets, class of labels the class R of all binary relations, and

$$\underline{Rel}(A,B) = \{ A \xrightarrow{R} B \mid A,B \in \mathcal{V} \text{ and } R \subseteq A \times B \}.$$

For each $A \in \mathcal{V}$, $Id_A : A \to A$ is just the identity function (a special case of a relation), and given $A \xrightarrow{R} B$ and $B \xrightarrow{G} C$, then $A \xrightarrow{R;G} C$ is sometimes denoted $A \xrightarrow{G \circ R} C$, the usual relation composition: $G \circ R = \{(a,c) \mid \exists b \in B \text{ s.t. } (a,b) \in R \wedge (b,c) \in G\}$

The Associativity and Identity axioms follow trivially from the properties of relation composition.

3. Before defining the third example we need to recall the notion of a commutative monoid.

Definition A commutative monoid is a triple $M = (M, +, 0)$ where M is a set, $+$ is a binary function $+ : M \times M \to M$, and $0 \in M$ satisfying for each $x, y, z \in M$ the laws:

1. Associativity: $(x+y)+z = x+(y+z)$
2. Commutativity: $x+y = y+z$
3. Unit: $x+0 = x$

Given commutative monoids $(M,+,0)$ and $(M',+',0')$ a monoid homomorphism is a function $f: M \to M'$ such that for each $x, y \in M$:

1. preserves addition: $f(x+y) = f(x) +' f(y)$
2. preserves 0: $f(0) = 0'$

For example, for $\mathbb{N}$ the natural numbers and $\mathbb{N}_{k+1} = \{0, \dots, k\}$, then the function $res_{k+1}: \mathbb{N} \ni n \longmapsto res(n, k+1) \in \mathbb{N}_{k+1}$ sends each $n \in \mathbb{N}$ to its reminder when divided by $k+1$ is a commutative monoid homomorphism

$$res_{k+1} : (\mathbb{N}, +, 0) \longrightarrow (\mathbb{N}_{k+1}, +_{k+1}, 0)$$

where $+_{k+1}$ is addition modulo $k+1$.

The category $\underline{CMon} = (\mathfrak{C}Mon, \mathfrak{C}MonHom, \underset{Mon}{\rightarrow})$ has class of objects the class $\mathfrak{C}Mon$ of all commutative monoids, as labels the class $\mathfrak{C}MonHom$ of all commutative monoid homomorphisms, and

$$\underline{CMon}((M,+,0),(M',+',0')) = \left\{ (M,+,0) \xrightarrow{f} (M',+',0') \;\middle|\; f:(M,+,0)\to(M',+',0') \text{ commutative monoid homomorphism} \right\}$$

This is a (large) category because:

(i) $1_M : (M,+,0) \to (M,+,0)$ is a commutative monoid homomorphism,

(ii) If $(M,+,0) \xrightarrow{f} (M',+',0')$ and $(M',+',0') \xrightarrow{g} (M'',+'',0'')$ are commutative monoid homomorphisms, so is $(M,+,0) \xrightarrow{g \circ f} (M'',+'',0'')$; (iii) Associativity and Identity hold.