# Propositional logic with short-circuit evaluation: a non-commutative and a commutative variant

Bergstra, J.A.; Ponse, A.; Staudt, D.J.C.

**Publication date**
2018

**Document Version**
Submitted manuscript

# Propositional logic with short-circuit evaluation: a non-commutative and a commutative variant

Jan A. Bergstra Alban Ponse Daan J.C. Staudt

Section Theory of Computer Science, Informatics Institute
Faculty of Science, University of Amsterdam

https://staff.science.uva.nl/{j.a.bergstra,a.ponse} https://www.daanstaudt.nl

**Abstract**

Short-circuit evaluation denotes the semantics of propositional connectives in which the second argument is evaluated only if the first argument does not suffice to determine the value of the expression. Short-circuit evaluation is widely used in programming, with sequential conjunction and disjunction as primitive connectives.

We study the question which logical laws axiomatize short-circuit evaluation under the following assumptions: compound statements are evaluated from left to right, each atom (propositional variable) evaluates to either true or false, and atomic evaluations can cause a side effect. The answer to this question depends on the kind of atomic side effects that can occur and leads to different "short-circuit logics". The basic case is FSCL (free short-circuit logic), which characterizes the setting in which each atomic evaluation can cause a side effect. We recall some main results and then relate FSCL to MSCL (memorizing short-circuit logic), where in the evaluation of a compound statement, the first evaluation result of each atom is memorized. MSCL can be seen as a sequential variant of propositional logic: atomic evaluations cannot cause a side effect and the sequential connectives are not commutative. Then we relate MSCL to SSCL (static short-circuit logic), the variant of propositional logic that prescribes short-circuit evaluation with commutative sequential connectives.

We present evaluation trees as an intuitive semantics for short-circuit evaluation, and simple equational axiomatizations for the short-circuit logics mentioned that use negation and the sequential connectives only.

*Keywords:* Non-commutative conjunction, conditional composition, sequential connectives, short-circuit evaluation, side effect

## Contents

# 1 Introduction

In this paper, we discern a fixed evaluation strategy to determine the truth of a propositional statement. We proceed from some very simple points of departure:

- Atoms (propositional variables) evaluate to either *true* or *false*, thus we exclude logics that comprise other truth values.
- The semantics of the binary propositional connectives (conjunction and disjunction) is determined by *short-circuit evaluation*: the second argument is evaluated only if the first argument does not suffice to determine the (evaluation) value of the expression.
- Once an atom in a compound expression is evaluated to a truth value, each next atomic evaluation of that atom evaluates to the same truth value. For example, if $a$ evaluates to *true*, then so does $a \wedge a$.

We consider conjunction as the primary connective and disjunction as a derived connective, and we write

$$\owedge \quad \text{and} \quad \ovee$$

for the case that these connectives *prescribe* short-circuit evaluation. This notation stems from [2], where the small circle indicates that the left argument must be evaluated first. Other notations are `&&` and `||` as used in programming, $\otimes$ and $\oplus$ from transaction logic (see, e.g. [1]), and $\triangle$ and $\triangledown$ from computability logic (see, e.g. [11]). However, we prefer the asymmetric symbols and we will henceforth refer to these as *sequential connectives*. Given a set of atoms (propositional variables), sequential propositions are built from atoms, sequential conjunction and disjunction as mentioned here, negation, and the constants $\mathsf{T}$ and $\mathsf{F}$ for the values *true* and *false*.

Short-circuit evaluation combines well with negation, and sequential (equational) variants of De Morgan's laws are valid, such as

$$\neg(x \owedge y) = \neg x \ovee \neg y.$$

We first recall *free short-circuit logic*, FSCL for short, and relate this to two variants of propositional logic with short-circuit evaluation. In FSCL, two sequential propositions are identified

if and only if they always have the same evaluation value under short-circuit evaluation. Here "always" refers to any possible assumption about the truth value of atoms in any evaluation state, and to the side effects that may occur in the evaluation process: we speak of an *atomic side effect* if the evaluation of an atom in a compound expression changes (influences) the evaluation result of the subsequent atoms that must be evaluated to determine value of the expression. FSCL is a logic for equational reasoning about sequential propositions that may have atomic side effects without any restriction. Stated differently, this logic is immune to all atomic side effects. For example, in FSCL the sequential proposition $a \mathbin{\curlywedge} a$ is not equivalent with $a$ or with $a \mathbin{\curlywedge} (a \mathbin{\curlyvee} a)$. Two typical laws of FSCL are $(x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z = x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)$ and $x \mathbin{\curlywedge} \mathsf{F} = \neg x \mathbin{\curlywedge} \mathsf{F}$.

In this paper we study two short-circuit logics that comprise FSCL:

**MSCL**, "memorizing short-circuit logic", is a logic for equational reasoning about sequential propositions with the property that atomic side effects do not occur: in the evaluation of a compound statement the first evaluation result of each atom is memorized. In this logic, the sequential connectives are not commutative, for example, $a \mathbin{\curlywedge} \mathsf{F}$ and $\mathsf{F} \mathbin{\curlywedge} a$ are not equivalent (the first sequential proposition requires evaluation of atom $a$, the second one does not). Typical laws of MSCL are $x \mathbin{\curlywedge} x = x$ and $x \mathbin{\curlywedge} (y \mathbin{\curlywedge} x) = x \mathbin{\curlywedge} y$.

**SSCL**, "static short-circuit logic", is the (equational) variant of propositional logic in which short-circuit evaluation is prescribed, thus the sequential connectives are taken to be commutative. Also this logic is based on the assumption that atomic side effects do not occur.

**Structure of the paper and main results.** In Section 2 we discuss evaluation trees, which model the evaluation of a sequential proposition and were defined in [15, 14]. We recall the main results on FSCL, in particular its equational axiomatization EqFSCL for closed terms.

In Section 3 we define memorizing evaluation trees by a transformation on the evaluation trees for FSCL, introduce EqMSCL as an equational axiomatization of their equality, and show that the axioms of EqFSCL are derivable from EqMSCL.

In Section 4 we recall the definitions of the short-circuit logics mentioned above. These definitions employ the *conditional* — a ternary connective introduced by Hoare in 1985 in [10] — as a hidden operator, and stem from [8, 6].

In Section 5 we prove that EqMSCL corresponds with MSCL in the sense that both define the same equational theory, and that both axiomatize equality of memorizing evaluation trees.

In Section 6 we define EqSSCL as the extension of EqMSCL with a commutativity axiom, and prove that EqSSCL is an equational axiomatization of SSCL. Then we show that both axiomatize equality of static evaluation trees as defined in [7]. Finally, we present four simple axioms for the conditional connective as an alternative for those in [10].

Section 7 contains some conclusions, in particular on viewing both MSCL and SSCL as variants of propositional logic.

**Notes.** 1. All derivability results in this paper were checked with the theorem prover *Prover9*, and all independence results were found with help of the tool *Mace4*, see [12] for both these tools. We added four appendices with detailed proofs of these results.

2. Considerable parts of the text below stem from [15, 8, 14]. Together with [14], this paper subsumes most of [8]. Two topics discussed in [8] and not in this paper are 'repetition-proof' and 'contractive' short-circuit logic; we will deal with these topics in a forthcoming paper.

## 2 Evaluation trees and axioms for short-circuit evaluation

In this section we summarize the main results of [14]: evaluation trees and an axiomatization of their equality are discussed

Given a non-empty set $A$ of atoms, we first define evaluation trees.

**Definition 2.1.** *The set $\mathcal{T}_A$ of **evaluation trees** over $A$ with leaves in $\{\mathsf{T},\mathsf{F}\}$ is defined inductively by*

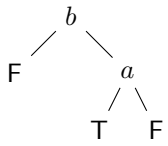
$$\mathsf{T} \in \mathcal{T}_A, \quad \mathsf{F} \in \mathcal{T}_A, \quad (X \trianglelefteq a \trianglerighteq Y) \in \mathcal{T}_A \text{ for any } X,Y \in \mathcal{T}_A \text{ and } a \in A.$$

*The operator $\_ \trianglelefteq a \trianglerighteq \_$ is called **tree composition over** $a$. In the evaluation tree $X \trianglelefteq a \trianglerighteq Y$, the root is represented by $a$, the left branch by $X$ and the right branch by $Y$.*

The leaves of an evaluation tree represent evaluation results (so we use the constants $\mathsf{T}$ and $\mathsf{F}$ for *true* and *false*). Next to the formal notation for evaluation trees we also use a more pictorial representation. For example, the tree

$$\mathsf{F} \trianglelefteq b \trianglerighteq (\mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F})$$

can be represented as follows, where $\trianglelefteq$ yields a left branch, and $\trianglerighteq$ a right branch:




(Picture 1)

In order to define a short-circuit semantics for negation and the sequential connectives, we first define the *leaf replacement* operator, 'replacement' for short, on trees in $\mathcal{T}_A$ as follows. For $X \in \mathcal{T}_A$, the replacement of $\mathsf{T}$ with $Y$ and $\mathsf{F}$ with $Z$ in $X$, denoted

$$X[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z]$$

is defined recursively by

$$\begin{aligned} \mathsf{T}[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z] &= Y, \\ \mathsf{F}[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z] &= Z, \\ (X_1 \trianglelefteq a \trianglerighteq X_2)[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z] &= X_1[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z] \trianglelefteq a \trianglerighteq X_2[\mathsf{T} \mapsto Y, \mathsf{F} \mapsto Z]. \end{aligned}$$

We note that the order in which the replacements of leaves of $X$ is listed is irrelevant and adopt the convention of not listing identities inside the brackets, e.g., $X[\mathsf{F} \mapsto Z] = X[\mathsf{T} \mapsto \mathsf{T}, \mathsf{F} \mapsto Z]$. By structural induction it follows that repeated replacements satisfy

$$X[\mathsf{T} \mapsto Y_1, \mathsf{F} \mapsto Z_1][\mathsf{T} \mapsto Y_2, \mathsf{F} \mapsto Z_2] = X[\mathsf{T} \mapsto Y_1[\mathsf{T} \mapsto Y_2, \mathsf{F} \mapsto Z_2],\ \mathsf{F} \mapsto Z_1[\mathsf{T} \mapsto Y_2, \mathsf{F} \mapsto Z_2]].$$

We define the set $\mathcal{S}_A$ of closed (sequential) propositional statements over $A$ by the following grammar:

$$P ::= a \mid \mathsf{T} \mid \mathsf{F} \mid \neg P \mid P \mathbin{\wedge\!\!\!\diagup} P \mid P \mathbin{\diagdown\!\!\!\vee} P,$$

where $a \in A$, $\mathsf{T}$ is a constant for the truth value *true*, $\mathsf{F}$ for *false*, and refer to its signature by

$$\Sigma_{\mathrm{SCL}}(A) = \{\mathbin{\wedge\!\!\!\diagup}, \mathbin{\diagdown\!\!\!\vee}, \neg, \mathsf{T}, \mathsf{F}, a \mid a \in A\}.$$

We interpret propositional statements in $\mathcal{S}_A$ as evaluation trees by a function *se* (abbreviating short-circuit evaluation).

$$\mathsf{F} = \neg\mathsf{T} \tag{F1}$$
$$x \mathbin{\curlyvee} y = \neg(\neg x \mathbin{\curlywedge} \neg y) \tag{F2}$$
$$\neg\neg x = x \tag{F3}$$
$$\mathsf{T} \mathbin{\curlywedge} x = x \tag{F4}$$
$$x \mathbin{\curlyvee} \mathsf{F} = x \tag{F5}$$
$$\mathsf{F} \mathbin{\curlywedge} x = \mathsf{F} \tag{F6}$$
$$(x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z = x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z) \tag{F7}$$
$$\neg x \mathbin{\curlywedge} \mathsf{F} = x \mathbin{\curlywedge} \mathsf{F} \tag{F8}$$
$$(x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} y = (x \mathbin{\curlyvee} \mathsf{T}) \mathbin{\curlywedge} y \tag{F9}$$
$$(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}) = (x \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F})) \mathbin{\curlywedge} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F})) \tag{F10}$$

Table 1: EqFSCL, a set of axioms for *se*-congruence

**Definition 2.2.** *The unary* ***short-circuit evaluation function*** $se : \mathcal{S}_A \to \mathcal{T}_A$ *is defined as follows, where* $a \in A$*:*

$$\begin{aligned} se(\mathsf{T}) &= \mathsf{T}, & se(\neg P) &= se(P)[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}], \\ se(\mathsf{F}) &= \mathsf{F}, & se(P \mathbin{\curlywedge} Q) &= se(P)[\mathsf{T} \mapsto se(Q)], \\ se(a) &= \mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F}, & se(P \mathbin{\curlyvee} Q) &= se(P)[\mathsf{F} \mapsto se(Q)]. \end{aligned}$$

The overloading of the symbol $\mathsf{T}$ in $se(\mathsf{T}) = \mathsf{T}$ will not cause confusion (and similarly for $\mathsf{F}$). As a simple example we derive the evaluation tree for $\neg b \mathbin{\curlywedge} a$:

$$se(\neg b \mathbin{\curlywedge} a) = se(\neg b)[\mathsf{T} \mapsto se(a)] = (\mathsf{F} \trianglelefteq b \trianglerighteq \mathsf{T})[\mathsf{T} \mapsto se(a)] = \mathsf{F} \trianglelefteq b \trianglerighteq (\mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F}),$$

which can be visualized as Picture 1 on page 4. Also, $se(\neg(b \mathbin{\curlyvee} \neg a)) = \mathsf{F} \trianglelefteq b \trianglerighteq (\mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F})$. An evaluation tree $se(P)$ represents short-circuit evaluation in a way that can be compared to the notion of a truth table for propositional logic in that it represents each possible evaluation of $P$. However, there are some important differences with truth tables: in $se(P)$, the sequentiality of $P$'s evaluation is represented, and the same atom may occur multiple times in $se(P)$.

**Definition 2.3.** *The binary relation* ***se-congruence****, notation* $=_{se}$*, is defined on* $\mathcal{S}_A$ *by*

$$P =_{se} Q \iff se(P) = se(Q).$$

In [15, 14] it is proved that the axioms in Table 1[1] constitute an equational axiomatization of *se*-congruence:

**Fact 2.4.** *For all* $P, Q \in \mathcal{S}_A$*,* $\mathrm{EqFSCL} \vdash P = Q \iff P =_{se} Q$*.*

This implies that the axioms in Table 1 axiomatize free short-circuit logic FSCL (defined in Section 4) for closed terms, and for this reason this set of axioms is named EqFSCL. Some comments on these axioms: (F1)-(F3) imply sequential versions of De Morgan's laws, and thus a

[1] In [15], the dual of axiom (F5) is used.

sequential variant of the duality principle. Axioms (F4)-(F6) define how the constants $\mathsf{T}$ and $\mathsf{F}$ interact with the sequential connectives, and axiom (F7) defines the associativity of $\between$.

Axiom (F8) defines a typical property of a logic that characterizes immunity for side effects: although it is the case that for each $P \in \mathcal{S}_A$, the evaluation result of $P \between \mathsf{F}$ is *false*, the evaluation of $P$ might also yield a side effect. However, the same side effect and evaluation result are obtained upon evaluation of $\neg P \between \mathsf{F}$.

Axiom (F9) expresses another property that concerns possible side effects: because the evaluation result of $P \between \mathsf{F}$ for each possible evaluation of the atoms in $P$ is *false*, $Q$ is always evaluated in $(P \between \mathsf{F}) \mathbin{\curlyvee} Q$ and determines the evaluation result, which is also the case in $(P \mathbin{\curlyvee} \mathsf{T}) \between Q$. Note that the evaluations of $P \mathbin{\curlyvee} \mathsf{T}$ and $P \between \mathsf{F}$ accumulate the same side effects, which perhaps is more easily seen if one replaces $Q$ by either $\mathsf{T}$ or $\mathsf{F}$.

Axiom (F10) defines a restricted form of right-distributivity of $\curlyvee$ and (by duality) of $\between$. This axiom holds because if $x$ evaluates to *true*, both sides further evaluate $y \mathbin{\curlyvee} (z \between \mathsf{F})$, and if $x$ evaluates to *false*, $z \between \mathsf{F}$ determines the further evaluation result (which is then *false*, and by axiom (F6), $y \mathbin{\curlyvee} (z \between \mathsf{F})$ is not evaluated in the right-hand side).

The dual of $P \in \mathcal{S}_A$, notation $P^{dl}$, is defined as follows (for $a \in A$):

$$\begin{array}{lll} \mathsf{T}^{dl} = \mathsf{F}, & a^{dl} = a, & (P \between Q)^{dl} = P^{dl} \mathbin{\curlyvee} Q^{dl}, \\ \mathsf{F}^{dl} = \mathsf{T}, & (\neg P)^{dl} = \neg P^{dl}, & (P \mathbin{\curlyvee} Q)^{dl} = P^{dl} \between Q^{dl}. \end{array}$$

The duality mapping $(\,)^{dl} : \mathcal{S}_A \to \mathcal{S}_A$ is an involution, that is, $(P^{dl})^{dl} = P$. Setting $x^{dl} = x$ for each variable $x$, the duality principle extends to equations, e.g., the dual of axiom (F7) is $(x \mathbin{\curlyvee} y) \mathbin{\curlyvee} z = x \mathbin{\curlyvee} (y \mathbin{\curlyvee} z)$. From (F1)-(F3) it immediately follows that EqFSCL satisfies the duality principle, that is, for all terms $s, t$ over $\Sigma_{\mathrm{SCL}}(A)$,

$$\mathrm{EqFSCL} \vdash s = t \quad \Longleftrightarrow \quad \mathrm{EqFSCL} \vdash s^{dl} = t^{dl}.$$

We conclude this section with some more properties of EqFSCL that were proved in [14].

**Fact 2.5.** *Let* $\mathrm{EqFSCL}^- = \mathrm{EqFSCL} \setminus \{(\mathrm{F1}), (\mathrm{F3})\}$. *Then*

$$\mathrm{EqFSCL}^- \setminus \{(\mathrm{F8}), (\mathrm{F10})\} \vdash (\mathrm{F1}), (\mathrm{F3}), \text{ and thus } \mathrm{EqFSCL}^- \vdash \mathrm{EqFSCL},$$

*and the axioms of* $\mathrm{EqFSCL}^-$ *are independent if* $A$ *contains at least two atoms.*

# 3 Evaluation trees and axioms for memorizing short-circuit evaluation

In this section memorizing evaluation trees and an axiomatization of their equality are introduced, as well as a congruence that identifies more than *se*-congruence.

A short-circuit evaluation is *memorizing* if in the evaluation of a compound statement the first evaluation result of each atom is memorized. Typically, the following sequential version of the absorption law holds under memorizing evaluations:

$$x \between (x \mathbin{\curlyvee} y) = x. \tag{Abs}$$

Equation (Abs) can be explained as follows: if $x$ evaluates to *false*, then $x \between (x \mathbin{\curlyvee} y)$ evaluates to *false* as a result of the evaluation of the left occurrence of $x$ (and $(x \mathbin{\curlyvee} y)$ is not evaluated); if $x$

evaluates to *true*, the second evaluation of $x$ in the subterm $(x \mathbin{\curlyvee} y)$ also results in *true* (because it is memorizing) and therefore $y$ is not evaluated.

A perhaps less obvious property of memorizing evaluations is the following:

$$(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z = (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z). \tag{Mem}$$

If $x$ evaluates to *true*, then $z$ determines the evaluation result of both expressions because the evaluation result of $x$ is memorized; if $x$ evaluates to *false*, the evaluation result of both expressions is determined by $y \mathbin{\curlywedge} z$ because the right disjunct $(x \mathbin{\curlywedge} z)$ also evaluates to *false*.

Below we define the *memorizing evaluation function* as a transformation on evaluation trees. This transformation implements the characteristic of memorizing evaluations starting at the root of an evaluation tree, and removes each second occurrence of a label $a$ according to its first evaluation result. Intuitively, memorizing evaluations are those of propositional logic, except that the sequential connectives are not commutative. As an example, $a \mathbin{\curlywedge} b$ and $b \mathbin{\curlywedge} a$ represent different evaluations, and hence are not equivalent.

**Definition 3.1.** *The unary* ***memorizing evaluation function***

$$mse : \mathcal{S}_A \to \mathcal{T}_A$$

*yields* ***memorizing evaluation trees*** *and is defined by*

$$mse(P) = m(se(P)).$$

*The auxiliary function $m : \mathcal{T}_A \to \mathcal{T}_A$ is defined as follows ($a \in A$):*

$$\begin{aligned} m(\mathsf{T}) &= \mathsf{T}, \\ m(\mathsf{F}) &= \mathsf{F}, \\ m(X \trianglelefteq a \trianglerighteq Y) &= m(L_a(X)) \trianglelefteq a \trianglerighteq m(R_a(Y)). \end{aligned}$$

*For $a \in A$, the auxiliary functions $L_a : \mathcal{T}_A \to \mathcal{T}_A$ ("Left a-reduction") and $R_a : \mathcal{T}_A \to \mathcal{T}_A$ ("Right a-reduction") are defined by*

$$\begin{aligned} L_a(\mathsf{T}) &= \mathsf{T}, \\ L_a(\mathsf{F}) &= \mathsf{F}, \\ L_a(X \trianglelefteq b \trianglerighteq Y) &= \begin{cases} L_a(X) & \text{if } b = a, \\ L_a(X) \trianglelefteq b \trianglerighteq L_a(Y) & \text{otherwise,} \end{cases} \end{aligned}$$

*and*

$$\begin{aligned} R_a(\mathsf{T}) &= \mathsf{T}, \\ R_a(\mathsf{F}) &= \mathsf{F}, \\ R_a(X \trianglelefteq b \trianglerighteq Y) &= \begin{cases} R_a(Y) & \text{if } b = a, \\ R_a(X) \trianglelefteq b \trianglerighteq R_a(Y) & \text{otherwise.} \end{cases} \end{aligned}$$

| | |
|---|---|
| $\mathsf{F} = \neg\mathsf{T}$ | (Neg) |
| $x \curlyvee y = \neg(\neg x \curlywedge \neg y)$ | (Or) |
| $\mathsf{T} \curlywedge x = x$ | (Tand) |
| $x \curlywedge (x \curlyvee y) = x$ | (Abs) |
| $(x \curlyvee y) \curlywedge z = (\neg x \curlywedge (y \curlywedge z)) \curlyvee (x \curlywedge z)$ | (Mem) |

Table 2: EqMSCL, a set of axioms for memorizing *se*-congruence

As an example we depict $se(a \curlywedge (b \curlywedge a))$ and the memorizing evaluation tree $mse(a \curlywedge (b \curlywedge a))$:


From a more general point of view, a memorizing evaluation tree is a *decision tree*, that is a labeled, rooted, binary tree with internal nodes labeled from $A$ and leaves labeled from $\{\mathsf{T}, \mathsf{F}\}$ such that for any path from the root to a leaf, the internal nodes receive distinct labels (cf. [13]).

Equality of memorizing evaluation trees defines a congruence on $\mathcal{S}_A$.

**Definition 3.2.** ***Memorizing*** *se***-congruence***, notation* $=^{\mathcal{S}}_{mse}$*, is defined on* $\mathcal{S}_A$ *by*

$$P =^{\mathcal{S}}_{mse} Q \iff mse(P) = mse(Q).$$

The superscript $\mathcal{S}$ in $=^{\mathcal{S}}_{mse}$ is used as a reference to $\mathcal{S}_A$ because later on we will consider a close variant of this congruence. In Section 5 we argue why $=^{\mathcal{S}}_{mse}$ is a congruence. Memorizing *se*-congruence identifies much more than *se*-congruence, but not as much as propositional logic, e.g., $\curlywedge$ and $\curlyvee$ are not commutative: $\mathsf{F} \curlywedge a \neq^{\mathcal{S}}_{mse} a \curlywedge \mathsf{F}$.

In Table 2 we present a set of equational axioms for $=^{\mathcal{S}}_{mse}$ and we call this set EqMSCL (this is a simplified version of EqMSCL as introduced in [8, 6]). One of our main results, proved in Section 5, is the following:

$$\text{For all } P, Q \in \mathcal{S}_A,\ \text{EqMSCL} \vdash P = Q \iff P =^{\mathcal{S}}_{mse} Q. \qquad \text{(Thm 5.9)}$$

To enhance readability, we renamed the EqFSCL-axioms used: (F1) → (Neg), (F2) → (Or), and (F4) → (Tand).

**Theorem 3.3.** *The axioms of* EqMSCL *are independent.*

*Proof.* By Theorem 6.3 (that states that a superset of EqMSCL is independent). □

The next theorem states that the EqFSCL-axioms are derivable from EqMSCL, and hence implies that *se*-congruence is subsumed by memorizing *se*-congruence.

**Theorem 3.4.** EqMSCL $\vdash$ EqFSCL.

*Proof.* With help of the theorem prover *Prover9*, see Appendix A.1. □

In the proof of Theorem 3.4, the EqFSCL-axioms are derived in a particular order, as to obtain useful intermediate results. Axiom (F3), that is $\neg\neg x = x$, is derived first, which justifies the use of the duality principle in subsequent derivations. For easy reference we mention here some particular results that are used in Section 5.

**Fact 3.5.** *The following equations are derivable from* EqMSCL *(and proved in Appendix A.1). Axioms* (Abs) *and* (F5)*, that is* $x \curlyvee \mathsf{F} = x$ *(easy to derive), imply* ***idempotence*** *of* $\curlywedge$*:*

$$x = x \curlywedge (x \curlyvee \mathsf{F}) = x \curlywedge x.$$

*Two auxiliary results (with simple proofs) that are repeatedly used are the following:*

$$x \curlywedge y = (x \curlywedge \mathsf{F}) \curlyvee (x \curlywedge y), \tag{Ar1}$$
$$x \curlyvee y = (\neg x \curlywedge y) \curlyvee x. \tag{Ar2}$$

*The following two intermediate results are used in the derivation of axiom* (F10) *and in Section 4 (note that with memorizing evaluations, these terms all express "*`if` $x$ `then` $y$ `else` $z$*"):*

$$(x \curlywedge y) \curlyvee (\neg x \curlywedge z) = (\neg x \curlyvee y) \curlywedge (x \curlyvee z), \tag{M1}$$
$$(x \curlywedge y) \curlyvee (\neg x \curlywedge z) = (\neg x \curlywedge z) \curlyvee (x \curlywedge y). \tag{M2}$$

A typical EqMSCL-consequence is $x \curlywedge (y \curlywedge x) = x \curlywedge y$ (cf. the last example on memorizing evaluation trees). First derive

$$\neg x \curlywedge \mathsf{F} \overset{(F8)}{=} x \curlywedge \mathsf{F} \overset{(Ar2)'}{=} (\neg x \curlyvee \mathsf{F}) \curlywedge x \overset{(F5)}{=} \neg x \curlywedge x, \tag{1}$$

where (Ar2)′ is the dual of (Ar2). Hence,

$$\begin{aligned}
x \curlywedge y &= (\neg x \curlyvee y) \curlywedge x && \text{by (Ar2)}' \\
&= (x \curlywedge (y \curlywedge x)) \curlyvee (\neg x \curlywedge x) && \text{by (Mem), (F3)} \\
&= (x \curlywedge (y \curlywedge x)) \curlyvee (\neg x \curlywedge \mathsf{F}) && \text{by (1)} \\
&= (\neg x \curlywedge \mathsf{F}) \curlyvee (x \curlywedge (y \curlywedge x)) && \text{by (M2)} \\
&= x \curlywedge (y \curlywedge x). && \text{by (F8), (Ar1)}
\end{aligned}$$

Another convenient result on EqMSCL, used in Section 5, is the following.

**Theorem 3.6.** *The following equations are derivable from* EqMSCL*, where* (LD) *abbreviates left-distributivity of* $\curlywedge$*.*

$$((x \curlywedge y) \curlyvee (\neg x \curlywedge z)) \curlywedge u = (x \curlywedge (y \curlywedge u)) \curlyvee (\neg x \curlywedge (z \curlywedge u)), \tag{M3}$$
$$x \curlywedge (y \curlyvee z) = (x \curlywedge y) \curlyvee (x \curlywedge z). \tag{LD}$$

*Proof.* With help of the theorem prover *Prover9*, see Appendix A.2. □

We end this section by mentioning two alternatives for EqMSCL.

**Proposition 3.7.** *Replacing axiom* (Mem) *in* EqMSCL *by* (M1)*, and either* (M3) *or*

$$((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z)) \mathbin{\curlywedge} u = (\neg x \mathbin{\curlywedge} (z \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u))$$

*(thus,* (M3)*'s commutative variant) constitutes an alternative for* EqMSCL*.*

Both these sets of axioms are independent (by *Mace4* [12]). With *Prover9* [12], derivations of (Ar1), (Ar2), (M2), and (Mem), respectively, are simple. In contrast to the proof of Theorem 3.4, a derivation of the associativity of $\curlywedge$ (axiom (F7)) is so simple that we show it here:

$$\begin{aligned}
(x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z &= ((\neg x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) \mathbin{\curlywedge} z && \text{by (Ar1), (F8)}\\
&= (\neg x \mathbin{\curlywedge} (\mathsf{F} \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) && \text{by (M3), (F3)}\\
&= ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z))) && \text{by (F6), (F8)}\\
&= x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z). && \text{by (Ar1)}
\end{aligned}$$

# 4 The conditional connective and three short-circuit logics

In this section we consider Hoare's *conditional*, a ternary connective that can be used for defining the sequential connectives of $\Sigma_{\mathrm{SCL}}(A) = \{\curlywedge, \curlyvee, \neg, \mathsf{T}, \mathsf{F}, a \mid a \in A\}$. Then we recall the definitions of free short-circuit logic (FSCL), memorizing short-circuit logic (MSCL), and static short-circuit logic (SSCL) that were published earlier.

In 1985, Hoare introduced the *conditional* ([10]), a ternary connective with notation

$$x \lhd y \rhd z.$$

A more common expression for the conditional $x \lhd y \rhd z$ is "`if` $y$ `then` $x$ `else` $z$", which emphasizes that $y$ is evaluated *first*, and depending on the outcome of this partial evaluation, either $x$ or $z$ is evaluated, which then determines the evaluation result. So, the evaluation strategy prescribed by this form of if-then-else is a prime example of a sequential evaluation strategy. In order to reason algebraically with conditional expressions, Hoare's 'operator like' notation $x \lhd y \rhd z$ seems indispensable. In [10] an equational axiomatization of propositional logic is provided that only uses the conditional. Furthermore it is described how the sequential connectives and negation are expressed in this set-up, although the sequential nature of the conditional's evaluation is not discussed in this paper. Hoare's axiomatization over the signature $\Sigma_{\mathrm{CP}}(A) = \{\_ \lhd \_ \rhd \_, \mathsf{T}, \mathsf{F}, a \mid a \in A\}$ consists of eleven axioms, including those in Table 3. In Section 6 we present a concise and simple alternative for this axiomatization.

We extend the definition of the function $se$ (Definition 2.2) to closed terms over $\Sigma_{\mathrm{CP}}(A)$ by adding the clause

$$se(P \lhd Q \rhd R) = se(Q)[\mathsf{T} \mapsto se(P), \mathsf{F} \mapsto se(R)]. \tag{2}$$

The four axioms in Table 3, named CP (for Conditional Propositions), establish a complete axiomatization of $se$-congruence over the signature $\Sigma_{\mathrm{CP}}(A)$:

$$\text{For all closed terms } P, Q \text{ over } \Sigma_{\mathrm{CP}}(A),\ \mathrm{CP} \vdash P = Q \iff se(P) = se(Q).$$

A simple proof of this fact is recorded in [7, Thm.2.11] (and repeated in [14]).

$$x \lhd \mathsf{T} \rhd y = x \tag{CP1}$$
$$x \lhd \mathsf{F} \rhd y = y \tag{CP2}$$
$$\mathsf{T} \lhd x \rhd \mathsf{F} = x \tag{CP3}$$
$$x \lhd (y \lhd z \rhd u) \rhd v = (x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v) \tag{CP4}$$

Table 3: The set CP of axioms for proposition algebra

With the conditional connective and the constants $\mathsf{T}$ and $\mathsf{F}$, the sequential connectives prescribing short-circuit evaluation are definable:

$$\neg x = \mathsf{F} \lhd x \rhd \mathsf{T}, \tag{3}$$
$$x \mathbin{\curlywedge} y = y \lhd x \rhd \mathsf{F}, \tag{4}$$
$$x \mathbin{\curlyvee} y = \mathsf{T} \lhd x \rhd y. \tag{5}$$

Note that these equations agree with the extension of the definition of the function $se$ in (2) above: $se(\neg P) = se(\mathsf{F} \lhd P \rhd \mathsf{T})$, $se(P \mathbin{\curlywedge} Q) = se(Q \lhd P \rhd \mathsf{F})$, and $se(P \mathbin{\curlyvee} Q) = se(\mathsf{T} \lhd P \rhd Q)$. Thus, the axioms in Table 3 combined with equations (3)-(5), say

$$\mathrm{CP}(\neg, \curlywedge, \curlyvee),$$

axiomatize equality of evaluation trees for closed terms over the enriched signature $\Sigma_{\mathrm{CP}}(A) \cup \Sigma_{\mathrm{SCL}}(A)$.

In order to capture memorizing evaluations, the following axiom is formulated in [4]:

$$x \lhd y \rhd (z \lhd u \rhd (v \lhd y \rhd w)) = x \lhd y \rhd (z \lhd u \rhd w) \tag{CPmem}$$

The axiom (CPmem) expresses that the first evaluation value of $y$ is memorized. We define

$$\mathrm{CP}_{mem} = \mathrm{CP} \cup \{(\mathrm{CPmem})\}.$$

In forthcoming proofs we use the fact that replacing the variable $y$ in axiom (CPmem) by $\mathsf{F} \lhd y \rhd \mathsf{T}$ and/or the variable $u$ by $\mathsf{F} \lhd u \rhd \mathsf{T}$ yields equivalent versions of this axiom:

$$(x \lhd y \rhd (z \lhd u \rhd v)) \lhd u \rhd w = (x \lhd y \rhd z) \lhd u \rhd w, \tag{CPmem1}$$
$$x \lhd y \rhd ((z \lhd y \rhd u) \lhd v \rhd w) = x \lhd y \rhd (u \lhd v \rhd w), \tag{CPmem2}$$
$$((x \lhd y \rhd z) \lhd u \rhd v) \lhd y \rhd w = (x \lhd u \rhd v) \lhd y \rhd w. \tag{CPmem3}$$

This follows easily with (CP4), (CP2), (CP1). Furthermore, if we replace $u$ by $\mathsf{F}$ in (CPmem), we find the *contraction law*

$$x \lhd y \rhd (v \lhd y \rhd w) = x \lhd y \rhd w, \tag{CPcon1}$$

and replacing $u$ by $\mathsf{T}$ in axiom (CPmem3) yields the symmetric contraction law

$$(x \lhd y \rhd z) \lhd y \rhd w = x \lhd y \rhd w. \tag{CPcon2}$$

With help of the tool *Mace4* [12] it easily follows that the axioms of $\mathrm{CP}_{mem}$ are independent, and therefore those of CP are also independent.

We write $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee)$ for the axioms of $\mathrm{CP}_{mem}$ extended with equations (3)-(5). An important property of $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee)$ is that the conditional connective can be expressed with the sequential connectives and negation. First, observe that it is trivial to derive

$$\neg x \curlywedge z = \mathsf{F} \lhd x \rhd z,$$

and hence

$$\begin{aligned}
(x \curlywedge y) \curlyvee (\neg x \curlywedge z) &= \mathsf{T} \lhd (y \lhd x \rhd \mathsf{F}) \rhd (\mathsf{F} \lhd x \rhd z) && \text{by (3)-(5) and the above} \\
&= (\mathsf{T} \lhd y \rhd (\mathsf{F} \lhd x \rhd z)) \lhd x \rhd (\mathsf{F} \lhd x \rhd z) && \text{by (CP4), (CP2)} \\
&= (\mathsf{T} \lhd y \rhd \mathsf{F}) \lhd x \rhd z && \text{by (CPmem1), (CPcon1)} \\
&= y \lhd x \rhd z. && \text{by (CP3)} \qquad (6)
\end{aligned}$$

In some cases it is convenient to use other equations:

$$(\neg x \curlywedge z) \curlyvee (x \curlywedge y) = y \lhd x \rhd z, \qquad (7)$$

$$(x \curlyvee z) \curlywedge (\neg x \curlyvee y) = y \lhd x \rhd z, \qquad (8)$$

$$(\neg x \curlyvee y) \curlywedge (x \curlyvee z) = y \lhd x \rhd z, \qquad (9)$$

which can all be proved from $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee)$ in a similar way.

In [6, 8] a set-up is provided for defining short-circuit logics in a generic way with help of the conditional by restricting the consequences of some CP-axiomatization extended with equation (3) (that is, $\neg x = \mathsf{F} \lhd x \rhd \mathsf{T}$) and equation (4) (i.e., $x \curlywedge y = y \lhd x \rhd \mathsf{F}$) to the signature $\Sigma_{\mathrm{SCL}}(A)$. So, the conditional connective is considered a *hidden operator*.

The definition below uses the export operator $\square$ of *Module algebra* [3] to express this in a concise way: in module algebra, $S \square X$ is the operation that exports the signature $S$ from module $X$ while declaring other signature elements hidden.

**Definition 4.1.** *A **short-circuit logic** is a logic that implies the consequences of the module expression*

$$\mathrm{SCL} = \{\mathsf{T}, \neg, \curlywedge\} \square (\mathrm{CP} \cup \{(3), (4)\}).$$

As a first example, $\mathrm{SCL} \vdash \neg\neg x = x$ can be proved as follows:

$$\begin{aligned}
\neg\neg x &= \mathsf{F} \lhd (\mathsf{F} \lhd x \rhd \mathsf{T}) \rhd \mathsf{T} && \text{by (3)} \\
&= (\mathsf{F} \lhd \mathsf{F} \rhd \mathsf{T}) \lhd x \rhd (\mathsf{F} \lhd \mathsf{T} \rhd \mathsf{T}) && \text{by (CP4)} \\
&= \mathsf{T} \lhd x \rhd \mathsf{F} && \text{by (CP2), (CP1)} \\
&= x. && \text{by (CP3)} \qquad (10)
\end{aligned}$$

In [6, 8], the following short-circuit logics were defined:

**Definition 4.2.** ***Free short-circuit logic*** (**FSCL**) *is the short-circuit logic that implies no other consequences than those of the module expression* SCL.

***Memorizing short-circuit logic*** (**MSCL**) *is the short-circuit logic that implies no other consequences than those of the module expression*

$$\{\mathsf{T}, \neg, \curlywedge\} \square (\mathrm{CP} \cup \{(3), (4), (\mathrm{CPmem})\}).$$

***Static short-circuit logic*** (**SSCL**) *is the short-circuit logic that implies no other consequences than those of the module expression*

$$\{\mathsf{T}, \neg, \mathbin{\text{ø}\!\wedge}\} \,\square\, (\mathrm{CP} \cup \{(3), (4), (\mathrm{CPmem})\} \cup \{\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}\}).$$

To enhance readability, we extend these short-circuit logics with the constant $\mathsf{F}$ and its defining equation (Neg), which is justified by the SCL-derivation

$$\begin{aligned} \mathsf{F} &= \mathsf{F} \lhd \mathsf{T} \rhd \mathsf{T} && \text{by (CP1)} \\ &= \neg\mathsf{T}, && \text{by (3)} \end{aligned} \tag{11}$$

and with the connective $\mathbin{\curlyvee}$ and its defining equation (Or) (thus, $x \mathbin{\curlyvee} y = \neg(\neg x \mathbin{\text{ø}\!\wedge} \neg y)$) by admitting equation (5) in SCL-derivations, that is, $x \mathbin{\curlyvee} y = \mathsf{T} \lhd x \rhd y$. This last extension is justified by

$$\begin{aligned} \neg(\neg x \mathbin{\text{ø}\!\wedge} \neg y) &= \mathsf{F} \lhd (\neg y \lhd (\mathsf{F} \lhd x \rhd \mathsf{T}) \rhd \mathsf{F}) \rhd \mathsf{T} && \text{by (3), (4)} \\ &= \mathsf{F} \lhd (\mathsf{F} \lhd x \rhd \neg y) \rhd \mathsf{T} && \text{by (CP4), (CP2), (CP1)} \\ &= (\mathsf{F} \lhd \mathsf{F} \rhd \mathsf{T}) \lhd x \rhd (\mathsf{F} \lhd \neg y \rhd \mathsf{T}) && \text{by (CP4)} \\ &= \mathsf{T} \lhd x \rhd y. && \text{by (CP2), (3), (10)} \end{aligned} \tag{12}$$

In [15, 14] the following results are proved:

$$\text{For all } P, Q \in \mathcal{S}_A,\ \mathrm{FSCL} \vdash P = Q \iff \mathrm{EqFSCL} \vdash P = Q \iff P =_{se} Q.$$

In the remainder of the paper we will prove similar results for MSCL and SSCL .

# 5 Completeness of EqMSCL

In this section we prove that EqMSCL and MSCL are equally strong, that is, both define the same equational theory. Furthermore, both constitute a complete axiomatization of memorizing *se*-congruence.

Given a signature $\Sigma$, we write

$$\mathbb{T}_{\Sigma,\mathcal{X}}$$

for the set of open terms over $\Sigma$ with variables in $\mathcal{X}$ (typical elements of $\mathcal{X}$ are $x, y, z, u, v, w$).

**Definition 5.1.** *Define the following two functions between sets of open terms:*

$f : \mathbb{T}_{\Sigma_{\mathbf{SCL}}(A),\mathcal{X}} \to \mathbb{T}_{\Sigma_{\mathbf{CP}}(A),\mathcal{X}}$ *is defined by*

$$\begin{aligned} f(bl) &= bl \quad \textit{for } bl \in \{\mathsf{T}, \mathsf{F}\}, & f(\neg t) &= \mathsf{F} \lhd f(t) \rhd \mathsf{T}, \\ f(a) &= a \quad \textit{for } a \in A, & f(t_1 \mathbin{\text{ø}\!\wedge} t_2) &= f(t_2) \lhd f(t_1) \rhd \mathsf{F}, \\ f(x) &= x \quad \textit{for } x \in \mathcal{X}, & f(t_1 \mathbin{\curlyvee} t_2) &= \mathsf{T} \lhd f(t_1) \rhd f(t_2). \end{aligned}$$

$g : \mathbb{T}_{\Sigma_{\mathbf{CP}}(A),\mathcal{X}} \to \mathbb{T}_{\Sigma_{\mathbf{SCL}}(A),\mathcal{X}}$ *is defined by*

$$\begin{aligned} g(bl) &= bl \quad \textit{for } bl \in \{\mathsf{T}, \mathsf{F}\}, & g(x) &= x \quad \textit{for } x \in \mathcal{X}, \\ g(a) &= a \quad \textit{for } a \in A, & g(t_1 \lhd t_2 \rhd t_3) &= (g(t_2) \mathbin{\text{ø}\!\wedge} g(t_1)) \mathbin{\curlyvee} (\neg g(t_2) \mathbin{\text{ø}\!\wedge} g(t_3)). \end{aligned}$$

**Lemma 5.2.** *For all* $t \in \mathbb{T}_{\Sigma_{\mathrm{SCL}}(A),\mathcal{X}}$, $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee) \vdash f(t) = t$.

*Proof.* By structural induction on $t$. □

**Lemma 5.3.** *For all* $s, t \in \mathbb{T}_{\Sigma_{\mathrm{CP}}(A),\mathcal{X}}$, $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee) \vdash s = t \;\Rightarrow\; \mathrm{CP}_{mem} \vdash s = t$.

*Proof.* In an equational proof of $\mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee) \vdash s = t$, each occurrence of one of the equations (3), (4), and (5) can be replaced by the corresponding $\mathbb{T}_{\Sigma_{\mathrm{CP}}(A),\mathcal{X}}$-identity. More precisely, any occurrence of $\neg x = \mathsf{F} \lhd x \rhd \mathsf{T}$ can be replaced by $\mathsf{F} \lhd x \rhd \mathsf{T} = \mathsf{F} \lhd x \rhd \mathsf{T}$, and similar for applications of (4) and (5).

Because $s$ and $t$ do not contain occurrences of $\neg$, $\curlywedge$, and $\curlyvee$, this yields an equational proof of $s = t$ in $\mathrm{CP}_{mem}$. □

**Lemma 5.4.** *For all* $s, t \in \mathbb{T}_{\Sigma_{\mathrm{CP}}(A),\mathcal{X}}$, $\mathrm{CP}_{mem} \vdash s = t \;\Rightarrow\; \mathrm{EqMSCL} \vdash g(s) = g(t)$.

*Proof.* The $g$-translation of each $\mathrm{CP}_{mem}$-axiom is derivable in EqMSCL.

Axiom (CP1). $g(x \lhd \mathsf{T} \rhd y) = (\mathsf{T} \curlywedge x) \curlyvee (\neg\mathsf{T} \curlywedge y) = x = g(x)$.

Axiom (CP2). $g(x \lhd \mathsf{F} \rhd y) = (\mathsf{F} \curlywedge x) \curlyvee (\neg\mathsf{F} \curlywedge y) = \mathsf{F} \curlyvee y = y = g(y)$.

Axiom (CP3). $g(\mathsf{T} \lhd x \rhd \mathsf{F}) = (x \curlywedge \mathsf{T}) \curlyvee (\neg x \curlywedge \mathsf{F}) = x \curlyvee (\neg x \curlywedge \mathsf{F}) \overset{(\mathrm{F8})}{=} x \curlyvee (x \curlywedge \mathsf{F}) \overset{(\mathrm{Abs})'}{=} x = g(x)$.

Axiom (CP4). We write "Assoc" for applications of associativity, and we use use (M1), (M2), (M3), and (LD) (see Fact 3.5).

$$
\begin{aligned}
&g(x \lhd (y \lhd z \rhd u) \rhd v)\\
&\quad = (g(y \lhd z \rhd u) \curlywedge x) \curlyvee (\neg g(y \lhd z \rhd u) \curlywedge v)\\
&\quad = ([(z \curlywedge y) \curlyvee (\neg z \curlywedge u)] \curlywedge x) \curlyvee (\neg[(z \curlywedge y) \curlyvee (\neg z \curlywedge u)] \curlywedge v)\\
&\quad = ([(z \curlywedge y) \curlyvee (\neg z \curlywedge u)] \curlywedge x) \curlyvee ([(\neg z \curlyvee \neg y) \curlywedge (z \curlyvee \neg u)] \curlywedge v)\\
&\quad = ([(z \curlywedge y) \curlyvee (\neg z \curlywedge u)] \curlywedge x) \curlyvee ([(z \curlywedge \neg y) \curlyvee (\neg z \curlywedge \neg u)] \curlywedge v) && \text{by (M1)}\\
&\quad = [(z \curlywedge (y \curlywedge x)) \curlyvee (\neg z \curlywedge (u \curlywedge x))] \curlyvee [(z \curlywedge (\neg y \curlywedge v)) \curlyvee (\neg z \curlywedge (\neg u \curlywedge v))], && \text{by (M3)}
\end{aligned}
$$

and

$$
\begin{aligned}
&g((x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v))\\
&\quad = (z \curlywedge g(x \lhd y \rhd v)) \curlyvee (\neg z \curlywedge g(x \lhd u \rhd v))\\
&\quad = (z \curlywedge ((y \curlywedge x) \curlyvee (\neg y \curlywedge v))) \curlyvee (\neg z \curlywedge ((u \curlywedge x) \curlyvee (\neg u \curlywedge v)))\\
&\quad = ([(z \curlywedge (y \curlywedge x)) \curlyvee (z \curlywedge (\neg y \curlywedge v))]) \curlyvee ([(\neg z \curlywedge (u \curlywedge x)) \curlyvee (\neg z \curlywedge (\neg u \curlywedge v))]) && \text{by (LD)}\\
&\quad = (z \curlywedge (y \curlywedge x)) \curlyvee [((z \curlywedge (\neg y \curlywedge v)) \curlyvee (\neg z \curlywedge (u \curlywedge x))) \curlyvee (\neg z \curlywedge (\neg u \curlywedge v))] && \text{by Assoc}\\
&\quad = (z \curlywedge (y \curlywedge x)) \curlyvee [((\neg z \curlywedge (u \curlywedge x)) \curlyvee (z \curlywedge (\neg y \curlywedge v))) \curlyvee (\neg z \curlywedge (\neg u \curlywedge v))] && \text{by (M2)}\\
&\quad = [(z \curlywedge (y \curlywedge x)) \curlyvee (\neg z \curlywedge (u \curlywedge x))] \curlyvee [(z \curlywedge (\neg y \curlywedge v)) \curlyvee (\neg z \curlywedge (\neg u \curlywedge v))]. && \text{by Assoc}
\end{aligned}
$$

Axiom (CPmem). As argued in Section 3, it is sufficient to derive axiom (CPmem1), that is,

$$(w \lhd y \rhd (z \lhd x \rhd u)) \lhd x \rhd v = (w \lhd y \rhd z) \lhd x \rhd v.$$

Derive

$$\begin{aligned} g((w \lhd y \rhd (z \lhd x \rhd u)) \lhd x \rhd v) &= (x \mathbin{\curlywedge} g(w \lhd y \rhd (z \lhd x \rhd u))) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} v) \\ &= (x \mathbin{\curlywedge} M) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} v), \\ g((w \lhd y \rhd z) \lhd x \rhd v) &= (x \mathbin{\curlywedge} g(w \lhd y \rhd z)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} v) && \text{by (6)} \\ &= (x \mathbin{\curlywedge} N) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} v), \end{aligned}$$

so it suffices to derive $x \mathbin{\curlywedge} M = x \mathbin{\curlywedge} N$. We use one auxiliary result and we write $(n)'$ for the dual version of equation $(n)$.

$$\begin{aligned} x \mathbin{\curlywedge} \mathsf{F} &= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlywedge} y && \text{by (F6), Assoc} \\ &= ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} \mathsf{F}) \mathbin{\curlywedge} y && \text{by (F5)} \\ &= ((\neg x \mathbin{\curlyvee} (\mathsf{F} \mathbin{\curlyvee} \mathsf{F})) \mathbin{\curlywedge} (x \mathbin{\curlyvee} \mathsf{F})) \mathbin{\curlywedge} y && \text{by (Mem)}' \\ &= (\neg x \mathbin{\curlywedge} x) \mathbin{\curlywedge} y && \text{by (F5)} \\ &= (x \mathbin{\curlywedge} \neg x) \mathbin{\curlywedge} y. && \text{by (M2)} \end{aligned} \tag{13}$$

Hence,

$$\begin{aligned} x \mathbin{\curlywedge} M &= x \mathbin{\curlywedge} g(w \lhd y \rhd (z \lhd x \rhd u)) \\ &= x \mathbin{\curlywedge} ((y \mathbin{\curlywedge} w) \mathbin{\curlyvee} (\neg y \mathbin{\curlywedge} ((x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} u)))) \\ &= x \mathbin{\curlywedge} ((\neg y \mathbin{\curlyvee} w) \mathbin{\curlywedge} (y \mathbin{\curlyvee} ((x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} u)))) && \text{by (M1)} \\ &= x \mathbin{\curlywedge} ((y \mathbin{\curlyvee} ((x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} u))) \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w)) && \text{by (M2)} \\ &= x \mathbin{\curlywedge} ((y \mathbin{\curlyvee} ((\neg x \mathbin{\curlywedge} u) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z))) \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w)) && \text{by (M2)} \\ &= [x \mathbin{\curlywedge} (y \mathbin{\curlyvee} ((\neg x \mathbin{\curlywedge} u) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z)))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by Assoc} \\ &= [(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} ((\neg x \mathbin{\curlywedge} u) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z)))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by (LD)} \\ &= [(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} (\neg x \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (x \mathbin{\curlywedge} z)))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by (LD)} \\ &= [(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (((x \mathbin{\curlywedge} \neg x) \mathbin{\curlywedge} u) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by Assoc, idempotence} \\ &= [(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by (13)} \\ &= [(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (\mathsf{F} \mathbin{\curlyvee} z))] \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by (LD)} \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlyvee} z)) \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w) && \text{by (F4)}', \text{(LD)} \\ &= x \mathbin{\curlywedge} ((y \mathbin{\curlyvee} z) \mathbin{\curlywedge} (\neg y \mathbin{\curlyvee} w)) && \text{by Assoc} \\ &= x \mathbin{\curlywedge} g(w \lhd y \rhd z) && \text{by (8)} \\ &= x \mathbin{\curlywedge} N. \end{aligned}$$

□

**Theorem 5.5.** *For all terms $s, t$ over $\Sigma_{\mathrm{SCL}}(A)$,* $\mathrm{EqMSCL} \vdash s = t \iff \mathrm{MSCL} \vdash s = t$.

*Proof.* ($\Rightarrow$) It suffices to derive the axioms of EqMSCL from MSCL.
Axiom (Neg). See (11).
Axiom (Or). This follows from (12).
Axiom (Tand). $\mathsf{T} \mathbin{\curlywedge} x = x \lhd \mathsf{T} \rhd \mathsf{F} = x$.
Axiom (Abs). $x \mathbin{\curlywedge} (x \mathbin{\curlyvee} y) = (\mathsf{T} \lhd x \rhd y) \lhd x \rhd \mathsf{F} \overset{(\mathrm{CPcon2})}{=} \mathsf{T} \lhd x \rhd \mathsf{F} = x$.

Axiom (Mem). Denote $(x \curlyvee y) \curlywedge z = (\neg x \curlywedge (y \curlywedge z)) \curlyvee (x \curlywedge z)$ by $L = R$. Then

$$\begin{aligned}
L &= z \lhd (\mathsf{T} \lhd x \rhd y) \rhd \mathsf{F} \\
&= z \lhd x \rhd (z \lhd y \rhd \mathsf{F}), && \text{by (CP4), (CP1)} \\
R &= \mathsf{T} \lhd ((z \lhd y \rhd \mathsf{F}) \lhd (\mathsf{F} \lhd x \rhd \mathsf{T}) \rhd \mathsf{F}) \rhd (z \lhd x \rhd \mathsf{F}) \\
&= \mathsf{T} \lhd (\mathsf{F} \lhd x \rhd (z \lhd y \rhd \mathsf{F})) \rhd (z \lhd x \rhd \mathsf{F}) && \text{by (CP4), (CP2), (CP1)} \\
&= [z \lhd x \rhd \mathsf{F}] \lhd x \rhd [\mathsf{T} \lhd (z \lhd y \rhd \mathsf{F}) \rhd (z \lhd x \rhd \mathsf{F})] && \text{by (CP4), (CP2)} \\
&= z \lhd x \rhd (\mathsf{T} \lhd (z \lhd y \rhd \mathsf{F}) \rhd \mathsf{F}) && \text{by (CPcon2), (CPmem2)} \\
&= z \lhd x \rhd (z \lhd y \rhd \mathsf{F}). && \text{by (CP3)}
\end{aligned}$$

($\Leftarrow$)

$$\begin{aligned}
\mathrm{MSCL} \vdash s = t &\Rightarrow \mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee) \vdash s = t && \text{by definition} \\
&\Rightarrow \mathrm{CP}_{mem}(\neg, \curlywedge, \curlyvee) \vdash f(s) = f(t) && \text{by Lemma 5.2} \\
&\Rightarrow \mathrm{CP}_{mem} \vdash f(s) = f(t) && \text{by Lemma 5.3} \\
&\Rightarrow \mathrm{EqMSCL} \vdash g(f(s)) = g(f(t)). && \text{by Lemma 5.4}
\end{aligned}$$

Hence, it suffices to derive for all $t \in \mathbb{T}_{\Sigma_{\mathrm{SCL}}(A),\mathcal{X}}$, $\mathrm{EqMSCL} \vdash g(f(t)) = t$. This follows easily by structural induction, we only show the inductive case $t = t_1 \curlyvee t_2$:

$$g(f(t_1 \curlyvee t_2)) \overset{\mathrm{IH}}{=} (t_1 \curlywedge \mathsf{T}) \curlyvee (\neg t_1 \curlywedge t_2) \overset{(\mathrm{M2}),(\mathrm{F5})'}{=} (\neg t_1 \curlywedge t_2) \curlyvee t_1 \overset{(\mathrm{Ar2})}{=} t_1 \curlyvee t_2.$$

□

Now, let $\mathcal{C}_A$ be the set of closed terms over $\Sigma_{\mathrm{CP}}(A)$.

**Definition 5.6.** *The binary relation $=^{\mathcal{C}}_{mse}$ on $\mathcal{C}_A$, **memorizing valuation congruence**, is defined by*

$$P =^{\mathcal{C}}_{mse} Q \iff mse^{\mathcal{C}}(P) = mse^{\mathcal{C}}(Q),$$

*where the function $mse^{\mathcal{C}} : \mathcal{C}_A \to \mathcal{T}_A$ is defined as in Definition 3.1, except that the function se is now defined as in* (2)*, that is,*

$$se(P \lhd Q \rhd R) = se(Q)[\mathsf{T} \mapsto se(P), \mathsf{F} \mapsto se(R)].$$

This definition stems from [7, Def.5.12]. In [7, Thm.5.14] we prove this completeness result:

$$\text{For all } P, Q \in \mathcal{C}_A,\ \mathrm{CP}_{mem} \vdash P = Q \iff P =^{\mathcal{C}}_{mse} Q. \tag{14}$$

This result depends on a non-trivial proof of the fact that $=^{\mathcal{C}}_{mse}$ is a congruence on $\mathcal{C}_A$.

**Lemma 5.7.** *For all $P, Q \in \mathcal{S}_A$, $\mathrm{EqMSCL} \vdash P = Q \Rightarrow P =^{\mathcal{S}}_{mse} Q$.*

*Proof.* We first show that $=^{\mathcal{S}}_{mse}$ is a congruence on $\mathcal{S}_A$. By structural induction, $se(P) = se(f(P))$ for all $P \in \mathcal{S}_A$, where the function $f$ is defined in Definition 5.1. Hence,

$$mse(P) = mse^{\mathcal{C}}(f(P)). \tag{15}$$

Assume $P_i =^{\mathcal{S}}_{mse} P'_i$ for $i \in \{1, 2\}$. By (15), $f(P_i) =^{\mathcal{C}}_{mse} f(P'_i)$, and because $=^{\mathcal{C}}_{mse}$ is a congruence on $\mathcal{C}_A$,

$$f(P_1 \curlywedge P_2) = f(P_2) \lhd f(P_1) \rhd \mathsf{F} =^{\mathcal{C}}_{mse} f(P'_2) \lhd f(P'_1) \rhd \mathsf{F} = f(P'_1 \curlywedge P'_2),$$

and thus $mse^{\mathcal{C}}(f(P_1 \mathbin{\curlywedge} P_2)) = mse^{\mathcal{C}}(f(P'_1 \mathbin{\curlywedge} P'_2))$. By (15), $mse(P_1 \mathbin{\curlywedge} P_2) = mse(P'_1 \mathbin{\curlywedge} P'_2)$, and thus $P_1 \mathbin{\curlywedge} P_2 =^{\mathcal{S}}_{mse} P'_1 \mathbin{\curlywedge} P'_2$. The remaining cases follow in a similar way.

Next, for all $P, Q \in \mathcal{S}_A$, if EqMSCL $\vdash P = Q$ then $P =^{\mathcal{S}}_{mse} Q$. This follows from the facts that $=^{\mathcal{S}}_{mse}$ is a congruence on $\mathcal{S}_A$ and that each closed instance of each axiom of EqMSCL satisfies $=^{\mathcal{S}}_{mse}$.[2] We only show this for axiom (Abs):

$$\begin{aligned}
mse(P \mathbin{\curlywedge} (P \mathbin{\curlyvee} Q)) &= mse^{\mathcal{C}}(f(P \mathbin{\curlywedge} (P \mathbin{\curlyvee} Q))) && \text{by (15)}\\
&= mse^{\mathcal{C}}((\mathsf{T} \lhd f(P) \rhd f(Q)) \lhd f(P) \rhd \mathsf{F}) && \text{by definition of } f\\
&= mse^{\mathcal{C}}(\mathsf{T} \lhd f(P) \rhd \mathsf{F}) && \text{by (14), (CPcon2)}\\
&= mse^{\mathcal{C}}(f(P)) && \text{by (14), (CP3)}\\
&= mse(P). && \text{by (15)}
\end{aligned}$$

□

**Theorem 5.8.** *For all* $P, Q \in \mathcal{S}_A$, MSCL $\vdash P = Q \iff P =^{\mathcal{S}}_{mse} Q$.

*Proof.* ($\Rightarrow$) If MSCL $\vdash P = Q$, then by Theorem 5.5, EqMSCL $\vdash P = Q$, and by Lemma 5.7, $P =^{\mathcal{S}}_{mse} Q$.

($\Leftarrow$) If $P =^{\mathcal{S}}_{mse} Q$, then by (15), $f(P) =^{\mathcal{C}}_{mse} f(Q)$. By (14), CP$_{mem} \vdash f(P) = f(Q)$, and thus CP$_{mem}(\neg, \mathbin{\curlywedge}, \mathbin{\curlyvee}) \vdash f(P) = f(Q)$, and thus by Lemma 5.2, CP$_{mem}(\neg, \mathbin{\curlywedge}, \mathbin{\curlyvee}) \vdash P = Q$. By definition of MSCL it follows that MSCL $\vdash P = Q$. □

Theorem 5.5 establishes that MSCL is axiomatized by the equational logic EqMSCL, and Theorem 5.8 establishes that MSCL axiomatizes equality of memorizing evaluation trees. Combining these results leads to a final theorem on this matter, which establishes that "memorizing short-circuit logic" as a concept is independent of the conditional connective, with memorizing evaluation trees at hand as a simple and natural semantics for representing memorizing short-circuit evaluations. This is fully in line with Fact 2.4 on "free short-circuit logic".

**Theorem 5.9.** *For all* $P, Q \in \mathcal{S}_A$, EqMSCL $\vdash P = Q \iff P =^{\mathcal{S}}_{mse} Q$.

# 6 Static short-circuit logic

Static short-circuit logic covers the case in which the sequential connectives are taken to be commutative. In this section we first discuss two axiomatizations, one that is an extension of EqMSCL with a commutativity axiom (Comm), and the one used in SSCL's definition (Def. 4.2). Then we discuss static evaluation trees and two completeness results. Finally, we provide four simple CP-equations that axiomatize static valuation congruence.

All axioms in Table 4 represent common laws for propositional logic when forgetting the prescribed short-circuit evaluation, except axiom (Mem). We name this set of axioms EqSSCL, and first prove some familiar laws without making use of axioms (Neg) and (Tand), and thus without using the constants $\mathsf{T}$ and $\mathsf{F}$.

[2] Without loss of generality it can be assumed that substitutions happen first in equational proofs (see, e.g., [9]).

$$\mathsf{F} = \neg\mathsf{T} \tag{Neg}$$
$$x \curlyvee y = \neg(\neg x \curlywedge \neg y) \tag{Or}$$
$$\mathsf{T} \curlywedge x = x \tag{Tand}$$
$$x \curlywedge (x \curlyvee y) = x \tag{Abs}$$
$$(x \curlyvee y) \curlywedge z = (\neg x \curlywedge (y \curlywedge z)) \curlyvee (x \curlywedge z) \tag{Mem}$$
$$x \curlywedge y = y \curlywedge x \tag{Comm}$$

Table 4: EqSSCL, a set of axioms for SSCL

**Theorem 6.1.** *The four* EqSSCL*-axioms* (Or), (Abs), (Mem), *and* (Comm) *imply idempotence and associativity of* $\curlywedge$ *and* $\curlyvee$*, the double negation shift* $\neg\neg x = x$ *(that is, axiom* (F3)*), and the equations*

$$(x \curlyvee \neg x) \curlywedge y = y, \tag{Tdef}$$
$$x \curlywedge (y \curlyvee z) = (x \curlywedge y) \curlyvee (x \curlywedge z). \tag{LD}$$

*Furthermore, if* $|A| \geq 2$*, these four axioms are independent.*

*Proof.* The mentioned derivabilities follow with help of the theorem prover *Prover9*, see Appendix A.3. For independence, see the proof of Theorem 6.3. □

This result is relevant because in EqSSCL the constants $\mathsf{T}$ and $\mathsf{F}$ are redundant (by equation (Tdef) and axiom (Comm), $x \curlyvee \neg x = y \curlyvee \neg y$). Note that in the setting without these constants, the duality principle is captured by axiom (Or) and the double negation shift.

By definition of EqSSCL we have the following theorem.

**Theorem 6.2.** EqSSCL $\vdash$ EqMSCL.

Furthermore, we have the following result, which implies that the axioms of EqMSCL are independent as well (cf. Theorem 3.3).

**Theorem 6.3.** *The axioms of* EqSSCL *are independent.*

*Proof.* With help of the tool *Mace4* [12], see Appendix A.4. □

We now return to static short-circuit logic SSCL as defined in Definition 4.2. In Table 5, the CP-axiom

$$\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F} \tag{CPs}$$

is added to $\mathrm{CP}_{mem}$ and the resulting set of axioms is denoted $\mathrm{CP}_s$. This set of axioms stems from [8, 6]. First, we formulate the analogue of Lemma 5.3 and establish a correspondence result for EqSSCL and SSCL.

**Lemma 6.4.** *For all* $s, t \in \mathbb{T}_{\Sigma_{\mathrm{CP}}(A),\mathcal{X}}$, $\mathrm{CP}_s(\neg, \curlywedge, \curlyvee) \vdash s = t \;\Rightarrow\; \mathrm{CP}_s \vdash s = t$.

*Proof.* See the proof of Lemma 5.3. □

$$x \lhd \mathsf{T} \rhd y = x \tag{CP1}$$
$$x \lhd \mathsf{F} \rhd y = y \tag{CP2}$$
$$\mathsf{T} \lhd x \rhd \mathsf{F} = x \tag{CP3}$$
$$x \lhd (y \lhd z \rhd u) \rhd v = (x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v) \tag{CP4}$$
$$x \lhd y \rhd (z \lhd u \rhd (v \lhd y \rhd w)) = x \lhd y \rhd (z \lhd u \rhd w) \tag{CPmem}$$
$$\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F} \tag{CPs}$$

Table 5: $\mathrm{CP}_s$, the set of CP-axioms used in SSCL's definition (Def. 4.1)

**Theorem 6.5.** *For all terms $s, t$ over $\Sigma_{\mathrm{SCL}}(A)$,* $\mathrm{EqSSCL} \vdash s = t \iff \mathrm{SSCL} \vdash s = t$.

*Proof.* ($\Rightarrow$) It suffices to derive the axioms of EqSSCL from SSCL, so by the proof of Theorem 5.5 we have to derive axiom (Comm). First derive

$$x \overset{\text{(CP2)}}{=} z \lhd \mathsf{F} \rhd x \overset{\text{(CPs)}}{=} z \lhd (\mathsf{F} \lhd y \rhd \mathsf{F}) \rhd x \overset{\text{(CP4)}}{=} (z \lhd \mathsf{F} \rhd x) \lhd y \rhd (z \lhd \mathsf{F} \rhd x) \overset{\text{(CP2)}}{=} x \lhd y \rhd x. \tag{16}$$

Hence

$$\begin{aligned}
x \mathbin{\curlywedge} y &= y \lhd x \rhd \mathsf{F} \\
&= y \lhd (x \lhd y \rhd x) \rhd \mathsf{F} && \text{by (16)} \\
&= ((\mathsf{T} \lhd y \rhd \mathsf{F}) \lhd x \rhd \mathsf{F}) \lhd y \rhd ((\mathsf{T} \lhd y \rhd \mathsf{F}) \lhd x \rhd \mathsf{F}) && \text{by (CP4), (CP1)} \\
&= (\mathsf{T} \lhd x \rhd \mathsf{F}) \lhd y \rhd (\mathsf{F} \lhd x \rhd \mathsf{F}) && \text{by (CPmem3), (CPmem2)} \\
&= x \lhd y \rhd \mathsf{F} && \text{by (CP1), (CPs)} \\
&= y \mathbin{\curlywedge} x.
\end{aligned}$$

($\Leftarrow$) Consider the functions $f$ and $g$ defined in Definition 5.1. We extend Lemma 5.4 to $\mathrm{CP}_s$:

$$\text{For all } s, t \in \mathbb{T}_{\Sigma_{\mathrm{CP}}(A), \mathcal{X}},\ \mathrm{CP}_s \vdash s = t \ \Rightarrow\ \mathrm{EqSSCL} \vdash g(s) = g(t). \tag{17}$$

The additional proof obligation is to show that the $g$-translation of the axiom (CPs) is derivable in EqSSCL (cf. Lemma 5.4):

$$\begin{aligned}
g(\mathsf{F} \lhd x \rhd \mathsf{F}) &= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} \mathsf{F}) \\
&= (\mathsf{F} \mathbin{\curlywedge} x) \mathbin{\curlyvee} (\mathsf{F} \mathbin{\curlywedge} \neg x) && \text{by (Comm)} \\
&= \mathsf{F} && \text{by (F6), (F5) (and Theorems 6.2, 3.4)} \\
&= g(\mathsf{F}).
\end{aligned}$$

We adapt the ($\Leftarrow$)-part of the proof of Theorem 5.5 to SSCL.

$$\begin{aligned}
\mathrm{SSCL} \vdash s = t \ &\Rightarrow\ \mathrm{CP}_s(\neg, \mathbin{\curlywedge}, \mathbin{\curlyvee}) \vdash s = t && \text{by definition} \\
&\Rightarrow\ \mathrm{CP}_s(\neg, \mathbin{\curlywedge}, \mathbin{\curlyvee}) \vdash f(s) = f(t) && \text{by Lemma 5.2} \\
&\Rightarrow\ \mathrm{CP}_s \vdash f(s) = f(t) && \text{by Lemma 6.4} \\
&\Rightarrow\ \mathrm{EqSSCL} \vdash g(f(s)) = g(f(t)). && \text{by (17)}
\end{aligned}$$

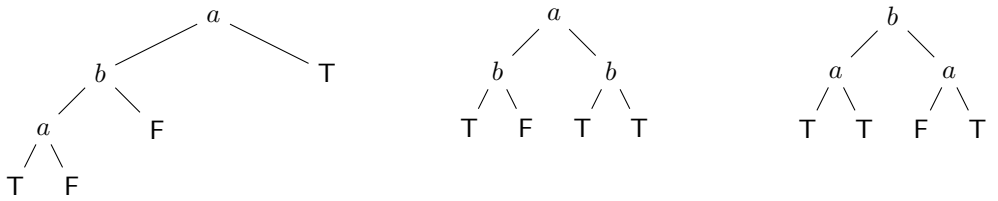
Hence, it suffices to show for all $t \in \mathbb{T}_{\Sigma_{\mathrm{SCL}}(A), \mathcal{X}}$, $\mathrm{EqSSCL} \vdash g(f(t)) = t$, and by $\mathrm{EqSSCL} \vdash \mathrm{EqMSCL}$ (Thm.6.2) this follows as in the ($\Leftarrow$)-part of the proof of Theorem 5.5. ∎

In [7], static evaluation trees for conditional propositions are defined with help of memorizing evaluation trees. The crux is that given a conditonal proposition $P$ and a finite set of atoms $A'$ that contains all atoms in $P$'s evaluation, the evaluation tree of $P$ is defined relative to an ordering of $A'$. We denote such an ordering as a string of length $|A'|$ that covers $A'$, for example, the orderings of $A' = \{a, b\}$ are denoted by $ab$ and $ba$. We write

$$A^u$$

for the set of strings representing all such orderings, and $\mathcal{S}_\sigma$ for the set of sequential propositions with atoms in $\sigma \in A^u$. Before defining the static evaluation function, we give an example.

**Example 6.6.** Let $P = \neg a \curlyvee (b \curlywedge a)$. We depict $se(P)$ at the left-hand side, and two static evaluation trees for $P$.




The two static evaluation trees correspond to the different ways in which one can present a (minimal) truth table for $P$, that is, the different possible orderings of the valuation values of the atoms occurring in $P$:

| $a$ | $b$ | $\neg a \curlyvee (b \curlywedge a)$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

| $b$ | $a$ | $\neg a \curlyvee (b \curlywedge a)$ |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | F |
| F | F | T |

The idea is that each proposition with atoms in $\{a, b\}$ has a static evaluation tree that is either of the form of the middle tree, or of the tree on the right, depending on which $\sigma \in A^u$ is chosen, and that the leaves represent the appropriate evaluation results. E.g., the leaves in the static evaluation trees for $\mathsf{F}$ and for $\mathsf{F} \curlywedge P$ are all $\mathsf{F}$. *End example.*

Because static evaluation trees do not necessary reflect the order of atomic evaluations, we do not take the trouble to define these directly for $\mathcal{S}_A$, but reuse their definition for $\mathcal{C}_A$, taken from [7, Def.6.13].[3] For $\sigma \in A^u$, let $\mathcal{C}_\sigma$ be the set of closed terms over $\Sigma_{\mathrm{CP}}(A)$ with atoms in $\sigma$.

**Definition 6.7.** *Let $\sigma \in A^u$. The unary **static evaluation function***

$$sse^{\mathcal{C}}_\sigma : \mathcal{C}_\sigma \to \mathcal{T}_A$$

*yields **static evaluation trees** and is defined as follows:*

$$sse^{\mathcal{C}}_\sigma(P) = mse^{\mathcal{C}}(\mathsf{T} \lhd E_\sigma \rhd P),$$

*with $mse^{\mathcal{C}}$ as in Definition 5.6, and $E_\sigma$ defined by $E_{a\rho} = E_\rho \lhd a \rhd E_\rho$ if $\sigma = a\rho$ for $a \in A$, and $E_\epsilon = \mathsf{F}$ with $\epsilon$ the empty string.*

[3] We come back to this point in Section 7.

As an example, the static evalution tree $sse^{\mathcal{C}}_{ab}(\mathsf{F}) = sse^{\mathcal{C}}_{ab}(\mathsf{F} \lhd a \rhd \mathsf{F}) = sse^{\mathcal{C}}_{ab}(\mathsf{F} \lhd b \rhd \mathsf{F})$ is depicted at the left-hand side, and $sse^{\mathcal{C}}_{ba}(\mathsf{F}) = sse^{\mathcal{C}}_{ba}(\mathsf{F} \lhd a \rhd \mathsf{F}) = sse^{\mathcal{C}}_{ba}(\mathsf{F} \lhd b \rhd \mathsf{F})$ is the other tree.


Static evaluation trees are perfect binary trees, where each level characterises the evaluation of a single atom.

**Definition 6.8.** *Let $\sigma \in A^u$. The binary relation $=^{\mathcal{C}}_{sse,\sigma}$ on $\mathcal{C}_\sigma$, **static valuation congruence over** $\sigma$, is defined by*

$$P =^{\mathcal{C}}_{sse,\sigma} Q \iff sse^{\mathcal{C}}_{\sigma}(P) = sse^{\mathcal{C}}_{\sigma}(Q).$$

This definition stems from [7, Def.6.14]. In [7, Thm.6.16] we prove this completeness result:

$$\text{Let } \sigma \in A^u. \text{ For all } P, Q \in \mathcal{C}_\sigma, \ \mathrm{CP}_s \vdash P = Q \iff P =^{\mathcal{C}}_{sse,\sigma} Q. \tag{18}$$

This result depends on a non-trivial proof of the fact that $=^{\mathcal{C}}_{sse,\sigma}$ is a congruence on $\mathcal{C}_\sigma$. We define the following variants of static evaluation trees and static valuation congruence for $\mathcal{S}_\sigma$.

**Definition 6.9.** *Let $\sigma \in A^u$. The unary **static evaluation function** $sse_\sigma : \mathcal{S}_\sigma \to \mathcal{T}_A$ is defined by $sse_\sigma(P) = sse^{\mathcal{C}}_{\sigma}(f(P))$, where $sse^{\mathcal{C}}_{\sigma}$ and $f$ are defined in Definitions 6.7 and 5.1.*

*The binary relation $=^{\mathcal{S}}_{sse,\sigma}$ on $\mathcal{S}_\sigma$, **static se-congruence over** $\sigma$, is defined on $\mathcal{S}_\sigma$ by*

$$P =^{\mathcal{S}}_{sse,\sigma} Q \iff f(P) =^{\mathcal{C}}_{sse,\sigma} f(Q).$$

Hence, the two trees in the example above are also the static evaluation trees $sse_{ab}(\mathsf{F}) = sse_{ab}(a \mathbin{\not\wedge} \mathsf{F}) = sse_{ab}(b \mathbin{\not\wedge} \mathsf{F})$ and $sse_{ba}(\mathsf{F}) = sse_{ba}(a \mathbin{\not\wedge} \mathsf{F}) = sse_{ba}(b \mathbin{\not\wedge} \mathsf{F})$, respectively.

**Theorem 6.10.** *Let $\sigma \in A^u$. For all $P, Q \in \mathcal{S}_\sigma$, $\mathrm{SSCL} \vdash P = Q \iff P =^{\mathcal{S}}_{sse,\sigma} Q$.*

*Proof.* By Lemma 5.2, it follows that for all $R \in \mathcal{S}_\sigma$, $\mathrm{CP}_s \cup \{(3), (4), (5)\} \vdash R = f(R)$. Hence, $\mathrm{SSCL} \vdash P = Q \iff \mathrm{CP}_s \cup \{(3), (4), (5)\} \vdash P = Q \iff \mathrm{CP}_s \cup \{(3), (4), (5)\} \vdash f(P) = f(Q) \iff \mathrm{CP}_s \vdash f(P) = f(Q)$, where the last implication $\Rightarrow$ follows from Lemma 6.4. By (18), the latter derivability holds if and only if $f(P) =^{\mathcal{C}}_{mse} f(Q)$, that is, $P =^{\mathcal{S}}_{mse} Q$. □

It is cumbersome, but not difficult to define static evaluation trees directly from memorizing evaluation trees: adapt Definition 6.9 by defining $D_{a\rho} = (a \mathbin{\not\wedge} \neg a) \mathbin{\not\vee} D_\rho$, $D_\epsilon = \mathsf{F}$, $sse_\sigma(P) = mse(D_\sigma \mathbin{\not\vee} P)$, and $P =^{\mathcal{S}}_{sse,\sigma} Q \iff sse_\sigma(P) = sse_\sigma(Q)$. This defines exactly the same static evaluation trees and relation $=^{\mathcal{S}}_{sse,\sigma}$, and thus provides a semantics for static short-circuit evaluations without use of the conditional connective. In this case, Theorem 6.10 can be proved in a similar way as Theorem 5.8 (which would then require the analogue of Lemma 5.7).

By Theorem 6.5, static short-circuit logic (SSCL) is axiomatized by the equational logic EqSSCL. By Theorem 6.10, SSCL axiomatizes equality of static evaluation trees. Thus, "static short-circuit logic" as a concept is independent of the conditional connective, and leads to the following completeness theorem.

$$x \lhd \mathsf{T} \rhd y = x \tag{CP1}$$
$$x \lhd \mathsf{F} \rhd y = y \tag{CP2}$$
$$(x \lhd y \rhd z) \lhd y \rhd \mathsf{F} = y \lhd x \rhd \mathsf{F} \tag{CP3s}$$
$$x \lhd (y \lhd z \rhd u) \rhd v = (x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v) \tag{CP4}$$

Table 6: An alternative set of CP-axioms for defining SSCL

**Theorem 6.11.** *Let $\sigma \in A^u$. For all $P, Q \in \mathcal{S}_\sigma$,* $\mathrm{EqSSCL} \vdash P = Q \iff P =^{\mathcal{S}}_{mse,\sigma} Q$.

Observe that this is again fully in line with Fact 2.4 on "free short-circuit logic" and Theorem 5.9 on "memorizing short-circuit logic".

We conclude this section with a few words on the definition of static short-circuit logic (Def. 4.1). In Table 6 we provide an alternative set of axioms for defining SSCL, thus for defining static valuation congruence. This axiomatization is independent (which easily follows with *Mace4* [12]), but is not a simple extension of CP or $\mathrm{CP}_{mem}$. Note that the axiom (CP3s) with $y = \mathsf{T}$ implies (CP3), and with $y = \mathsf{F}$ the axiom (CPs). A proof of one of the axioms (CPmem1) or (CPmem3) by *Prover9* [12] is relatively simple (with the option kbo); for the first one, a convenient intermediate result is

```
f(f(f(x,y,z),u,v),y,0)=f(f(x,u,v),y,0),
```

that is,

$$((x \lhd y \rhd z) \lhd u \rhd v) \lhd y \rhd \mathsf{F} = (x \lhd u \rhd v) \lhd y \rhd \mathsf{F},$$

and adding this as a fifth axiom yields a comprehensible proof of (CPmem1).

However, finding a more simple axiomatization of static valuation congruence is not a purpose of this paper: the axiomatization $\mathrm{CP}_s$ in Table 5 is sufficiently simple and expresses the fundamental intuitions in an appropriate way. Reasons to present the axiomatization in Table 6 are its independence (contrary to $\mathrm{CP}_s$, see below) and, of course, its striking simplicity (cf. [10]).

**Proposition 6.12.** $\mathrm{CP}_s \setminus \{(\mathrm{CP1})\} \vdash (\mathrm{CP1})$*, and the axioms of* $\mathrm{CP}_s \setminus \{(\mathrm{CP1})\}$ *are independent.*

*Proof.* First derive

$$\begin{aligned} x \lhd y \rhd (z \lhd u \rhd y) &= x \lhd y \rhd (z \lhd u \rhd (\mathsf{T} \lhd y \rhd \mathsf{F})) && \text{by (CP3)} \\ &= x \lhd y \rhd (z \lhd u \rhd \mathsf{F}), && \text{by (CPmem)} \end{aligned} \tag{19}$$

$$\begin{aligned} x \lhd \mathsf{T} \rhd y &= x \lhd \mathsf{T} \rhd (\mathsf{T} \lhd y \rhd \mathsf{F}) && \text{by (CP3)} \\ &= x \lhd \mathsf{T} \rhd (\mathsf{T} \lhd y \rhd \mathsf{T}) && \text{by (19)} \\ &= x \lhd \mathsf{T} \rhd \mathsf{T}. && \text{by (16)} \end{aligned} \tag{20}$$

Hence,

$$\begin{aligned} x \lhd \mathsf{T} \rhd y &= x \lhd \mathsf{T} \rhd x && \text{by (20)} \\ &= x. && \text{by (16)} \end{aligned}$$

The independence of $\mathrm{CP}_s \setminus \{(\mathrm{CP1})\}$ follows easily with help of the tool *Mace4* [12], where one atom is needed to show the independence of axiom (CP4) (recall that $A \neq \emptyset$). □

# 7 Conclusions

In [4] we introduced 'proposition algebra', which is based on Hoare's conditional $x \lhd y \rhd z$ and the constants $\mathsf{T}$ and $\mathsf{F}$. We defined a number of varieties of so-called *valuation algebras* in order to capture different semantics for the evaluation of conditional statements, and provided axiomatizations for the resulting valuation congruences: CP (four axioms) characterizes the least identifying valuation congruence we consider, and the extension $\mathrm{CP}_{mem}$ (one extra axiom) characterizes the most identifying valuation congruence below "sequential propositional logic". Static valuation congruence can be axiomatized by adding the axiom $\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}$ to $\mathrm{CP}_{mem}$, and can be seen as a characterization of (sequential) propositional logic.

In [5, 6] we introduced an alternative valuation semantics for proposition algebra in the form of *Hoare-McCarthy algebras* (HMA's) that is more elegant than the semantical framework provided in [4]: HMA-based semantics has the advantage that one can define a valuation congruence without first defining the valuation *equivalence* it is contained in.

In [7], following the approach of Staudt in [15], we defined evaluation trees as a more simple and direct semantics for proposition algebra and proved several completeness results for the valuation congruences mentioned above.

In [8] we introduced "short-circuit logic" as defined here (Def. 4.1 and Def. 4.2). In [14], we dealt with the case of free short-circuit logic (FSCL), as is summarized in Section 2.

In this paper we establish a setting in which memorizing short-circuit logic MSCL and static short-circuit logic SSCL can be understood and used without any reference to (or dependence on) the conditional connective.

From this perspective, MSCL can be seen as the equational logic defined by EqMSCL and with equality of memorizing evaluation trees as a simple semantics. MSCL can also be viewed as a short-circuited, operational variant of propositional logic: decisive for the meaning of a sequential proposition is the *process* of its sequential evaluation, as is clearly demonstrated by its memorizing evaluation tree, which also explains why the sequential connectives are taken to be non-commutative and why the constants $\mathsf{T}$ and $\mathsf{F}$ are not definable (and thus included). It is important to realize that a number of familiar properties hold in MSCL:

- The duality principle, the double negation shift, and associativity of the sequential connectives (all of these hold in FSCL).
- Idempotence of the sequential connectives, and

$$x \mathbin{\curlywedge} (y \mathbin{\curlywedge} x) = x \mathbin{\curlywedge} y, \qquad \text{(see page 9)}$$
$$x \mathbin{\curlywedge} (y \mathbin{\curlyvee} z) = (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z). \qquad \text{left-distributivity (LD)}$$

  (none of these hold in FSCL).

Some perhaps less familiar properties of MSCL, none of which hold in FSCL, are the following two characterizations of "$\texttt{if } x \texttt{ then } y \texttt{ else } z$":

$$(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) = (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z), \qquad \text{(M1)}$$
$$(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) = (\neg x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y), \qquad \text{(M2)}$$

and the right-distributivity of $\curlywedge$ over "$\texttt{if } x \texttt{ then } y \texttt{ else } z$", that is

$$(\texttt{if } x \texttt{ then } y \texttt{ else } z) \mathbin{\curlywedge} u = \texttt{if } x \texttt{ then } (y \mathbin{\curlywedge} u) \texttt{ else } (z \mathbin{\curlywedge} u),$$

which is characterized by

$$((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z)) \mathbin{\curlywedge} u = (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (z \mathbin{\curlywedge} u)). \qquad \text{(M3)}$$

Also, (M1) and the dual of (M3) imply right-distributivity of $\curlyvee$ over "`if` $x$ `then` $y$ `else` $z$":

$$((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z)) \mathbin{\curlyvee} u = (x \mathbin{\curlywedge} (y \mathbin{\curlyvee} u)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (z \mathbin{\curlyvee} u)).$$

Likewise, we can view SSCL as the equational logic defined by EqSSCL, and with equality of static evaluation trees as its semantics.[4] However, it is questionable whether equality of static evaluation trees is a useful semantics for SSCL (or EqSSCL), despite the interest of short-circuit connectives and short-circuit evaluation in propositional logic. Consider for example the identity $a \mathbin{\curlywedge} b = b \mathbin{\curlywedge} a$, which implies that the associated static evaluation trees should be considered equal. So, this either requires a transformation of *se*-evaluation trees according to an ordering of a fixed set of atoms (that contains $a$ and $b$), which may not agree with the evaluation order of atoms, or a non-intuitive equivalence relation between (ordinary) evaluation trees that does not respect this evaluation order. The same problem occurs in the case of expressions with the conditional and their static evaluation trees: the mismatch is that $b \lhd a \rhd \mathsf{F}$ models a sequential, short-circuited evaluation of $a \wedge b$, while the (necessary) identification $b \lhd a \rhd \mathsf{F} = a \lhd b \rhd \mathsf{F}$ declares the sequential nature of this evaluation irrelevant.

We conclude with some comments on the differences between MSCL and SSCL. First, the constant $\mathsf{T}$ is not definable in MSCL, but in SSCL it is definable by $x \mathbin{\curlyvee} \neg x$ (cf. Theorem 6.1). Next, short-circuit evaluation and *full evaluation* (prescribed by $\mathbin{\curlywedge\!\!\!\bullet}$, see [14, 15]) do not coincide in MSCL, but they do in SSCL:

$$x \mathbin{\curlywedge\!\!\!\bullet} y \stackrel{def}{=} (x \mathbin{\curlyvee} (y \mathbin{\curlywedge} \mathsf{F})) \mathbin{\curlywedge} y = (x \mathbin{\curlyvee} (\mathsf{F} \mathbin{\curlywedge} y)) \mathbin{\curlywedge} y = x \mathbin{\curlywedge} y.$$

Furthermore, in both MSCL and SSCL, the number of semantically different formulas is bounded by a function on $|A|$. This is an essential difference with short-circuit logics that identify less, such as FSCL. For $|A| = n$ (recall $n > 0$), the number of memorizing evaluation trees is $T_n = n(T_{n-1})^2 + 2$ with $T_0 = 2$ (so the first few values are $6, 74, 16430$),[5] and for $\sigma = a_1 a_2 ... a_n \in A^u$, the number of static evaluation trees over $\sigma$ is $2^{(2^n)}$. We finally note that the complexity of deciding satisfiability for both MSCL and SSCL is NP-complete (see [16, 4]). All in all, taking short-circuit evaluation and the absence of atomic side effects as points of departure, we think that MSCL provides a more natural view on (sequential) propositional logic than SSCL does.

**Related work.** In this paper we focused on the intrinsic properties of the sequential connectives in the setting of memorizing and static short-circuit evaluation, and we have not yet any specific applications in mind. Nevertheless, we mention a few areas of potentially related research. First, decision trees on Boolean variables as discussed in for example [13] are memorizing evaluation trees. Secondly, other notations for the sequential connectives $\curlywedge$ and $\curlyvee$ with memorizing interpretation are $\triangle$ and $\triangledown$ from computability logic (see, e.g. [11]), and $\otimes$ and $\oplus$ from transaction logic (see, e.g. [1]), there called *serial* connectives. However, MSCL is just a part of both these logics and it is questionable whether its axiomatization or semantics are of any relevance.

**Future work / Challenging questions.** With respect to the proof of Theorem 3.4, that is, EqMSCL $\vdash$ EqFSCL, find a shorter and more comprehensible proof of associativity. Alternatively, find another equational axiomatization for MSCL that is short and simple, uses only three variables, and admits a simple proof of this theorem.

[4] There is an innocent difference between the definition of static evaluation trees used in this paper (Def. 6.7) and its origin [7, Def.6.13]: the $\sigma$'s in the current definition are reversed, which we view as more natural.

[5] See `http://www.gzbjzb.com/oeis.org/A065410`.

# References


[1] Basseda, R. and Kifer, M. (2015). Planning with Regression Analysis in Transaction Logic. In: ten Cate, B. and Mileo, A. (eds.), RR 2015: Web Reasoning and Rule Systems. LNCS 9209, pages 45-60, 2015. Springer. DOI: 10.1007/978-3-319-22002-4_5.

[2] Bergstra, J.A., Bethke, I., and Rodenburg, P.H. (1995). A propositional logic with 4 values: true, false, divergent and meaningless. *Journal of Applied Non-Classical Logics*, 5(2):199-218.

[3] Bergstra, J.A., Heering, J., and Klint, P. (1990). Module algebra. *Journal of the ACM*, 37(2):335-372.

[4] Bergstra, J.A. and Ponse, A. (2011). Proposition algebra. *ACM Transactions on Computational Logic*, Vol. 12, No. 3, Article 21 (36 pages).

[5] Bergstra, J.A. and Ponse, A. (2010). On Hoare-McCarthy algebras. Available at `http://arxiv.org/abs/1012.5059` [cs.LO].

[6] Bergstra, J.A. and Ponse, A. (2012). Proposition algebra and short-circuit logic. In: Arbab, F. and Sirjani, M. (eds.), Proceedings of the 4th International Conference on Fundamentals of Software Engineering (FSEN 2011, Tehran), LNCS 7141, pages 15-31, Springer-Verlag.

[7] Bergstra, J.A. and Ponse, A. (2015). Evaluation trees for proposition algebra. Available at `http://arxiv.org/abs/1504.0832` [cs.LO].

[8] Bergstra, J.A., Ponse, A., and Staudt, D.J.C. (2013). Short-circuit logic. Available at `arXiv:1010.3674v4` [cs.LO,math.LO]. (First version appeared in 2010.)

[9] Groote, J.F (1990). A new strategy for proving omega-completeness applied to process algebra. In Baeten, J.C.M. and Klop, J.W. (eds.), *Theories of Concurrency: Unification and Extension* (CONCUR 1990, Amsterdam), LNCS 458, pages 314-331. Springer-Verlag.

[10] Hoare, C.A.R. (1985). A couple of novelties in the propositional calculus. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 31(2):173-178.

[11] Japaridze, G. (2008). Sequential operators in computability logic. *Information and Computation*, 206:1443-1475.

[12] McCune, W. (2008). The GUI: Prover9 and Mace4 with a Graphical User Interface. Prover9-Mace4-v05B.zip (March 14, 2008). Available at `https://www.cs.unm.edu/~mccune/prover9/gui/v05.html`.

[13] Moret, B.M.E. (1982). Decision trees and diagrams. *Computing Surveys*, 14(4):593-623. DOI: 10.1145/356893.356898.

[14] Ponse, A. and Staudt, D.J.C. (2018). An independent axiomatisation for free short-circuit logic. *Journal of Applied Non-Classical Logics*, 28(1):35-71. Online available. DOI: 10.1080/11663081.2018.1448637. (Also available at `arXiv:1707.05718v2` [cs.LO].)

[15] Staudt, D.J.C. (2012). Completeness for Two Left-Sequential Logics. MSc. thesis Logic, University of Amsterdam (May 2012). Available at `arXiv:1206.1936v1` [cs.LO].

[16] Veld, S.L. in 't (2014). Satisfiability of Short Circuit Logic. BSc. thesis Mathematics and Computer Science, University of Amsterdam (July 2014). Available at `arXiv:1510.05162v1` [cs.LO].

# A Detailed proofs

## A.1 A proof of Theorem 3.4

**Theorem 3.4.** EqMSCL $\vdash$ EqFSCL.

*Proof.* With help of the theorem prover *Prover9* [12]. We derive the EqFSCL-axioms in a particular order, as to obtain useful intermediate results. Recall that $(n)'$ represents the dual of equation $(n)$.

**Axiom** (F3)**.** First derive

$$\mathsf{T} \curlyvee x \overset{(\mathrm{F4})}{=} \mathsf{T} \curlywedge (\mathsf{T} \curlyvee x) \overset{(\mathrm{Abs})}{=} \mathsf{T}. \tag{21}$$

Hence,

$$\begin{aligned} x &= (\mathsf{T} \curlyvee \mathsf{T}) \curlywedge x && \text{by (F4), (21)} \\ &= (\mathsf{F} \curlywedge (\mathsf{T} \curlywedge x)) \curlyvee (\mathsf{T} \curlywedge x) && \text{by (Mem), (F1)} \\ &= (\mathsf{F} \curlywedge x) \curlyvee x, && \text{by (F4)} \end{aligned} \tag{22}$$

and

$$\begin{aligned} \neg(\mathsf{F} \curlywedge \neg x) &= \neg(\neg\mathsf{T} \curlywedge \neg x) && \text{by (F1)} \\ &= \mathsf{T} \curlyvee x && \text{by (F2)} \\ &= \mathsf{T}. && \text{by (21)} \end{aligned} \tag{23}$$

Hence, $\neg(\mathsf{F} \curlywedge x) \overset{(22)}{=} \neg(\mathsf{F} \curlywedge ((\mathsf{F} \curlywedge x) \curlyvee x)) \overset{(\mathrm{F2})}{=} \neg(\mathsf{F} \curlywedge \neg(\neg(\mathsf{F} \curlywedge x) \curlywedge \neg x)) \overset{(23)}{=} \mathsf{T}$, and thus

$$\begin{aligned} z &= (\mathsf{T} \curlyvee y) \curlywedge z && \text{by (F4), (21)} \\ &= (\mathsf{F} \curlywedge (y \curlywedge z)) \curlyvee (\mathsf{T} \curlywedge z) && \text{by (Mem), (F1)} \\ &= \neg(\neg(\mathsf{F} \curlywedge (y \curlywedge z)) \curlywedge \neg z) && \text{by (F4), (F2)} \\ &= \neg(\mathsf{T} \curlywedge \neg z) && \text{by } \neg(\mathsf{F} \curlywedge x) = \mathsf{T} \\ &= \neg\neg z. && \text{by (F4)} \end{aligned} \tag{F3}$$

**Intermediate result 1 - Duality.** By axioms (F1)-(F3) the duality principle holds.

**Axiom** (F6)**.** $\mathsf{F} \curlywedge x = \mathsf{F}$ by $(21)'$.

**Axiom** (F5)**.** Instantiate (Mem) with $x = \mathsf{F}$ and $y = \mathsf{T}$, and apply $\neg\mathsf{F} = \mathsf{T}$ and (F4), (F6):

$$z = \mathsf{T} \curlywedge z \overset{(\mathrm{F4})'}{=} (\mathsf{F} \curlyvee \mathsf{T}) \curlywedge z \overset{(\mathrm{Mem})}{=} (\mathsf{T} \curlywedge (\mathsf{T} \curlywedge z)) \curlyvee (\mathsf{F} \curlywedge z) \overset{(\mathrm{F4}),(\mathrm{F6})}{=} z \curlyvee \mathsf{F}. \tag{F5}$$

**Intermediate result 2 - Idempotence.** By axiom (F5), $x = x \curlywedge (x \curlyvee \mathsf{F}) \overset{(\mathrm{Abs})}{=} x \curlywedge x$.

**Axiom** (F8)**.** We derive the dual equation. First derive

$$\begin{aligned} x \curlyvee \mathsf{T} &= (x \curlyvee \mathsf{T}) \curlywedge \mathsf{T} && \text{by (F4)} \\ &= (\neg x \curlywedge (\mathsf{T} \curlywedge \mathsf{T})) \curlyvee (x \curlywedge \mathsf{T}) && \text{by (Mem)} \\ &= \neg x \curlyvee x, && \text{by (F4)} \end{aligned} \tag{24}$$

and

$$\neg x \curlyvee \mathsf{T} = x \curlyvee \neg x. \qquad \text{by (24), (F3)} \tag{25}$$

Hence

$$\begin{aligned} x \barwedge \mathsf{T} &= (\neg x \barwedge x) \barwedge \mathsf{T} && \text{by (24), (F4)} \\ &= (x \barwedge (x \barwedge \mathsf{T})) \barwedge (\neg x \barwedge \mathsf{T}) && \text{by (Mem)} \\ &= x \barwedge \neg x && \text{by (F4), idempotence} \\ &= \neg x \barwedge \mathsf{T}. && \text{by (25)} \qquad (\mathrm{F8})' \end{aligned}$$

**Intermediate result 3 - four auxiliary results.**

$$\begin{aligned} x \barwedge y &= (x \barwedge \mathsf{F}) \barwedge y && \text{by (F5)} \\ &= (\neg x \barwedge (\mathsf{F} \barwedge y)) \barwedge (x \barwedge y) && \text{by (Mem)} \\ &= (x \barwedge \mathsf{F}) \barwedge (x \barwedge y). && \text{by (F6), (F8)} \qquad (\mathrm{Ar1}) \end{aligned}$$

$$\begin{aligned} x \barwedge y &= (x \barwedge y) \barwedge \mathsf{T} && \text{by (F5)}' \\ &= (\neg x \barwedge y) \barwedge x. && \text{by (Mem)} \qquad (\mathrm{Ar2}) \end{aligned}$$

$$\begin{aligned} x \barwedge y &= (x \barwedge \mathsf{T}) \barwedge (x \barwedge y) && \text{by (Ar1)}' \\ &= (\neg x \barwedge (x \barwedge y)) \barwedge (x \barwedge (x \barwedge y)) && \text{by (Mem)} \\ &= (\neg x \barwedge (x \barwedge y)) \barwedge x && \text{by (Abs)} \\ &= x \barwedge (x \barwedge y). && \text{by (Ar2)} \qquad (\mathrm{Ar3}) \end{aligned}$$

$$\begin{aligned} x \barwedge y &= (\neg x \barwedge y) \barwedge x && \text{by (Ar2)} \\ &= (\neg x \barwedge (\neg x \barwedge y)) \barwedge x && \text{by (Ar3)}' \\ &= x \barwedge (\neg x \barwedge y). && \text{by (Ar2)} \qquad (\mathrm{Ar4}) \end{aligned}$$

**Axiom** (F9)**.** First derive

$$\begin{aligned} (x \barwedge \mathsf{T}) \barwedge \mathsf{F} &= (\neg x \barwedge \mathsf{F}) \barwedge (x \barwedge \mathsf{F}) && \text{by (Mem), (F4)} \\ &= (x \barwedge \mathsf{F}) \barwedge (x \barwedge \mathsf{F}) && \text{by (F8)} \\ &= x \barwedge \mathsf{F}. && \text{by idempotence} \qquad (26) \end{aligned}$$

Hence,

$$\begin{aligned} (x \barwedge \mathsf{T}) \barwedge y &= ((x \barwedge \mathsf{T}) \barwedge \mathsf{F}) \barwedge ((x \barwedge \mathsf{T}) \barwedge y) && \text{by (Ar1)} \\ &= (x \barwedge \mathsf{F}) \barwedge ((x \barwedge \mathsf{T}) \barwedge y) && \text{by (26)} \\ &= (x \barwedge \mathsf{F}) \barwedge (\neg(x \barwedge \mathsf{F}) \barwedge y) && \text{by (F8)}' \\ &= (x \barwedge \mathsf{F}) \barwedge y. && \text{by (Ar4)} \qquad (\mathrm{F9}) \end{aligned}$$

**Intermediate result 4 - three more auxiliary results.**

$$(x \barwedge \mathsf{F}) \barwedge y = x \barwedge \mathsf{F} \qquad (\mathrm{Ar5})$$

$$x \barwedge (y \barwedge x) = x \barwedge y \qquad (\mathrm{Ar6})$$

$$(x \barwedge y) \barwedge x = x \barwedge y \qquad (\mathrm{Ar7})$$

First derive

$$\begin{aligned} (x \barwedge \mathsf{F}) \barwedge \mathsf{F} &= \neg(x \barwedge \mathsf{F}) \barwedge \mathsf{F} && \text{by (F8)} \\ &= (\neg x \barwedge \mathsf{T}) \barwedge \mathsf{F} \\ &= \neg x \barwedge \mathsf{F} && \text{by (26)} \\ &= x \barwedge \mathsf{F}, && \text{by (F8)} \qquad (27) \end{aligned}$$

hence

$$\begin{aligned}
(x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} y &= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} y) && \text{by (Ar3)}' \\
&= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} (((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} y)) && \text{by (Ar1)} \\
&= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} y)) && \text{by (27)} \\
&= x \mathbin{\not\wedge} \mathsf{F}. && \text{by (Abs)} \qquad \text{(Ar5)}
\end{aligned}$$

$$\begin{aligned}
x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x) &= (x \mathbin{\not\wedge} x) \mathbin{\not\wedge} (y \mathbin{\not\wedge} x) && \text{by idempotence} \\
&= ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} x) \mathbin{\not\wedge} (y \mathbin{\not\wedge} x) && \text{by (Ar1), idempotence} \\
&= (\neg(x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} (x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x))) \mathbin{\not\vee} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} (y \mathbin{\not\wedge} x)) && \text{by (Mem)} \\
&= ((\neg x \mathbin{\not\vee} \mathsf{T}) \mathbin{\not\wedge} (x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x))) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (Ar5)} \\
&= ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} (x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x))) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (F9), (F8)} \\
&= (x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x)) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (Ar1)} \\
&= (x \mathbin{\not\wedge} (y \mathbin{\not\wedge} x)) \mathbin{\not\vee} (\neg x \mathbin{\not\wedge} x) && \text{by (24)}' \\
&= (\neg x \mathbin{\not\vee} y) \mathbin{\not\wedge} x && \text{by (Mem)} \\
&= x \mathbin{\not\wedge} y. && \text{by (Ar2)}' \qquad \text{(Ar6)}
\end{aligned}$$

$$\begin{aligned}
(x \mathbin{\not\wedge} y) \mathbin{\not\wedge} x &= (x \mathbin{\not\wedge} y) \mathbin{\not\wedge} (x \mathbin{\not\wedge} (x \mathbin{\not\wedge} y)) && \text{by (Ar6)} \\
&= (x \mathbin{\not\wedge} y) \mathbin{\not\wedge} (x \mathbin{\not\wedge} y) && \text{by (Ar3)}' \\
&= x \mathbin{\not\wedge} y. && \text{by idempotence} \qquad \text{(Ar7)}
\end{aligned}$$

**Axiom** (F7)**.** We use the following auxiliary results:

$$\begin{aligned}
(x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z &= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) && (28) \\
(x \mathbin{\not\vee} y) \mathbin{\not\wedge} (y \mathbin{\not\wedge} z) &= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} (y \mathbin{\not\wedge} z) && (29) \\
\neg x \mathbin{\not\vee} (y \mathbin{\not\wedge} z) &= \neg x \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) && (35)
\end{aligned}$$

and derive associativity of $\not\wedge$ as follows:

$$\begin{aligned}
(x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z &= (x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) && \text{by (28)} \\
&= (x \mathbin{\not\vee} (x \mathbin{\not\wedge} y)) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) && \text{by (29)} \\
&= x \mathbin{\not\wedge} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) && \text{by (Abs)}' \\
&= (\neg x \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z)) \mathbin{\not\wedge} x && \text{by (Ar2)}' \\
&= (\neg x \mathbin{\not\vee} (y \mathbin{\not\wedge} z)) \mathbin{\not\wedge} x && \text{by (35)} \\
&= x \mathbin{\not\wedge} (y \mathbin{\not\wedge} z). && \text{by (Ar2)}' \qquad \text{(F7)}
\end{aligned}$$

We derive the above auxiliary results in order:

$$\begin{aligned}
(x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z) &= ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z)) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (Ar7)}' \\
&= ((x \mathbin{\not\vee} \mathsf{T}) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z)) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (F9)} \\
&= ((\neg x \mathbin{\not\vee} \mathsf{T}) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z)) \mathbin{\not\vee} (x \mathbin{\not\wedge} \mathsf{F}) && \text{by (F8)}' \\
&= (\neg(x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} ((x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z)) \mathbin{\not\vee} ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\wedge} z) && \text{by (Ar5)} \\
&= ((x \mathbin{\not\wedge} \mathsf{F}) \mathbin{\not\vee} (x \mathbin{\not\wedge} y)) \mathbin{\not\wedge} z && \text{by (Mem)} \\
&= (x \mathbin{\not\wedge} y) \mathbin{\not\wedge} z. && \text{by (Ar1)} \qquad (28)
\end{aligned}$$

$$
\begin{aligned}
(x \curlyvee y) \curlywedge (y \curlywedge z) &= (\neg x \curlywedge (y \curlywedge (y \curlywedge z))) \curlyvee (x \curlywedge (y \curlywedge z)) && \text{by (Mem)} \\
&= (\neg x \curlywedge (y \curlywedge z)) \curlyvee (x \curlywedge (y \curlywedge z)) && \text{by (Ar3)}' \\
&= (x \curlyvee \mathsf{T}) \curlywedge (y \curlywedge z) && \text{by (Mem)} \\
&= (x \curlywedge \mathsf{F}) \curlyvee (y \curlywedge z), && \text{by (F9)} \qquad (29)
\end{aligned}
$$

The remaining auxiliary results lead to (35):

$$
\begin{aligned}
(x \curlyvee y) \curlywedge (x \curlyvee z) &= (\neg x \curlywedge (y \curlywedge (x \curlyvee z))) \curlyvee (x \curlywedge (x \curlyvee z)) && \text{by (Mem)} \\
&= (\neg x \curlywedge (y \curlywedge (x \curlyvee z))) \curlyvee x && \text{by (Abs)} \\
&= x \curlyvee (y \curlywedge (x \curlyvee z)). && \text{by (Ar2)} \qquad (30)
\end{aligned}
$$

$$
\begin{aligned}
x \curlyvee (\neg x \curlyvee y) &= (\neg x \curlywedge (\neg x \curlyvee y)) \curlyvee x && \text{by (Ar2)} \\
&= \neg x \curlyvee x && \text{by (Abs)} \\
&= (\neg x \curlywedge \neg x) \curlyvee x && \text{by idempotence} \\
&= x \curlyvee \neg x. && \text{by (Ar2)} \qquad (31)
\end{aligned}
$$

$$
\begin{aligned}
x \curlyvee ((x \curlywedge z) \curlyvee y) &= x \curlyvee ((\neg x \curlyvee (z \curlyvee y)) \curlywedge (x \curlyvee y)) && \text{by (Mem)} \\
&= (x \curlyvee (\neg x \curlyvee (z \curlyvee y))) \curlywedge (x \curlyvee y) && \text{by (30)} \\
&= (x \curlyvee \neg x) \curlywedge (x \curlyvee y) && \text{by (31)} \\
&= x \curlyvee (\neg x \curlywedge (x \curlyvee y)) && \text{by (30)} \\
&= x \curlyvee (x \curlyvee y) && \text{by (Ar4)} \\
&= x \curlyvee y. && \text{by (Ar3)} \qquad (32)
\end{aligned}
$$

$$
\begin{aligned}
x \curlyvee y &= x \curlyvee ((x \curlywedge z) \curlyvee y) && \text{by (32)} \\
&= x \curlyvee ((x \curlywedge z) \curlyvee (y \curlyvee (x \curlywedge z))) && \text{by (Ar6)}' \\
&= x \curlyvee (y \curlyvee (x \curlywedge z)). && \text{by (32)} \qquad (33)
\end{aligned}
$$

$$
\begin{aligned}
x \curlyvee (y \curlywedge z) &= x \curlyvee (\neg x \curlywedge (y \curlywedge z)) && \text{by (Ar4)} \\
&= x \curlyvee ((\neg x \curlywedge (y \curlywedge z)) \curlyvee (x \curlywedge z)) && \text{by (33)} \\
&= x \curlyvee ((x \curlyvee y) \curlywedge z). && \text{by (Mem)} \qquad (34)
\end{aligned}
$$

$$
\begin{aligned}
\neg x \curlyvee (y \curlywedge z) &= \neg x \curlyvee ((\neg x \curlyvee y) \curlywedge z) && \text{by (34)} \\
&= \neg x \curlyvee ((\neg x \curlyvee (x \curlywedge y)) \curlywedge z) && \text{by (Ar4)} \\
&= \neg x \curlyvee ((x \curlywedge y) \curlywedge z). && \text{by (34)} \qquad (35)
\end{aligned}
$$

We write "Assoc" for (repeated) applications of associativity of $\curlywedge$ and $\curlyvee$.

**Intermediate result 5.** In order to derive axiom (F10) we use the following two intermediate results:

$$(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) = (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z) \tag{M1}$$
$$(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) = (\neg x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y) \tag{M2}$$

Equation (M1).

$$\begin{aligned}
(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) &= (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)) && \text{by (ir3)}' \\
&= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} (x \mathbin{\curlyvee} z))) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)) && \text{by (33)}' \\
&= (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z). && \text{by (Mem)} \qquad \text{(M1)}
\end{aligned}$$

Equation (M2). First derive

$$\begin{aligned}
(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z &= (x \mathbin{\curlyvee} y) \mathbin{\curlywedge} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z) && \text{by (Ar3)}' \\
&= (x \mathbin{\curlyvee} (x \mathbin{\curlyvee} y)) \mathbin{\curlywedge} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z) && \text{by (Ar3)} \\
&= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z), && \text{by (29)} \qquad (36)
\end{aligned}$$

$$\begin{aligned}
(x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} y &= (\neg x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} y && \text{by (F8)} \\
&= (\neg x \mathbin{\curlyvee} \mathsf{T}) \mathbin{\curlywedge} y && \text{by (F9)} \\
&= (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} y) && \text{by (Mem), (F4)} \\
&= (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} y) && \text{by (M1)} \\
&= (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} y)) \mathbin{\curlywedge} (x \mathbin{\curlyvee} y) && \text{by idempotence} \\
&= (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} y, && \text{by (Mem)}' \qquad (37)
\end{aligned}$$

$$\begin{aligned}
(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z &= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z) && \text{by (36)} \\
&= (x \mathbin{\curlywedge} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z)) \mathbin{\curlyvee} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z) && \text{by (37)} \\
&= ((x \mathbin{\curlywedge} (x \mathbin{\curlyvee} y)) \mathbin{\curlywedge} z) \mathbin{\curlyvee} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z) && \text{by Assoc} \\
&= (x \mathbin{\curlywedge} z) \mathbin{\curlyvee} ((x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z). && \text{by (Abs)} \qquad (38)
\end{aligned}$$

Hence,

$$\begin{aligned}
(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z) &= (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z) && \text{by (M1)} \\
&= (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z) \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} y)) && \text{by (Ar6)} \\
&= (\neg x \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z) \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} y)) && \text{by (Ar4)} \\
&= (\neg x \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) \mathbin{\curlywedge} ((\neg x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) && \text{by (M1)}' \\
&= (\neg x \mathbin{\curlywedge} z) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y). && \text{by (38)}' \qquad \text{(M2)}
\end{aligned}$$

**Axiom** (F10)**.** First derive

$$\begin{aligned}
(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} z &= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z) && \text{by (Mem)} \\
&= (x \mathbin{\curlyvee} (y \mathbin{\curlywedge} z)) \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} z). && \text{by (M1), (F3)} \qquad (39)
\end{aligned}$$

Hence,

$$\begin{aligned}
(x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}) &= (x \mathbin{\curlywedge} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}))) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (z \mathbin{\curlywedge} \mathsf{F})) && \text{by (39)}' \\
&= (x \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F})) \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}))) && \text{by (M1), (M2)} \\
&= (x \mathbin{\curlyvee} [(z \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlywedge} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}))]) \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F}))) && \text{by (Ar5)} \\
&= (x \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F})) \mathbin{\curlywedge} (y \mathbin{\curlyvee} (z \mathbin{\curlywedge} \mathsf{F})). && \text{by (39)} \qquad \text{(F10)}
\end{aligned}$$

□

## A.2 A proof of Theorem 3.6

**Theorem 3.6.** The following equations are derivable from EqMSCL, where (LD) abbreviates left-distributivity.

$$((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z)) \mathbin{\curlywedge} u = (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (z \mathbin{\curlywedge} u)), \tag{M3}$$

$$x \mathbin{\curlywedge} (y \mathbin{\curlyvee} z) = (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z). \tag{LD}$$

*Proof.* With help of the theorem prover *Prover9* [12].

**Equation** (M3)**.** First derive

$$\begin{aligned} x \mathbin{\curlywedge} (y \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z) \mathbin{\curlywedge} u)) &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} (x \mathbin{\curlyvee} z))) \mathbin{\curlywedge} u && \text{by Assoc} \\ &= (x \mathbin{\curlywedge} y) \mathbin{\curlywedge} u && \text{by } (33)' \\ &= x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u). && \text{by Assoc} \end{aligned} \tag{40}$$

Hence,

$$\begin{aligned} ((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} z)) \mathbin{\curlywedge} u &= ((\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)) \mathbin{\curlywedge} u && \text{by (M1)} \\ &= (\neg x \mathbin{\curlyvee} y) \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z) \mathbin{\curlywedge} u) && \text{by Assoc} \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z) \mathbin{\curlywedge} u))) \mathbin{\curlyvee} \\ &\quad (\neg x \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z)) \mathbin{\curlywedge} u) && \text{by (Mem)} \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} ((x \mathbin{\curlyvee} z)) \mathbin{\curlywedge} u) && \text{by (40)} \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} ((\neg x \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)) \mathbin{\curlywedge} u) && \text{by Assoc} \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} ((\neg x \mathbin{\curlywedge} z) \mathbin{\curlywedge} u) && \text{by } (\text{Ar4})' \\ &= (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} u)) \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (z \mathbin{\curlywedge} u)). && \text{by Assoc} \end{aligned} \tag{M3}$$

**Equation** (LD)**.** First derive

$$\begin{aligned} x \mathbin{\curlyvee} (y \mathbin{\curlyvee} z) &= (x \mathbin{\curlyvee} y) \mathbin{\curlyvee} z && \text{by Assoc} \\ &= ((x \mathbin{\curlyvee} y) \mathbin{\curlyvee} x) \mathbin{\curlyvee} z && \text{by } (\text{Ar7})' \\ &= x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (x \mathbin{\curlyvee} z)), && \text{by Assoc} \end{aligned} \tag{41}$$

$$\begin{aligned} \neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (x \mathbin{\curlywedge} z)) &= \neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (\neg x \mathbin{\curlyvee} (x \mathbin{\curlywedge} z))) && \text{by (41)} \\ &= \neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (\neg x \mathbin{\curlyvee} z)) && \text{by (Ar4)} \\ &= \neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} z). && \text{by (41)} \end{aligned} \tag{42}$$

Hence,

$$\begin{aligned} x \mathbin{\curlywedge} (y \mathbin{\curlyvee} z) &= (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} z)) \mathbin{\curlywedge} x && \text{by } (\text{Ar2})' \\ &= (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} z)) \mathbin{\curlywedge} (x \mathbin{\curlyvee} (x \mathbin{\curlywedge} z)) && \text{by } (\text{Abs})' \\ &= (\neg x \mathbin{\curlyvee} (y \mathbin{\curlyvee} (x \mathbin{\curlywedge} z))) \mathbin{\curlywedge} (x \mathbin{\curlyvee} (x \mathbin{\curlywedge} z)) && \text{by (42)} \\ &= (x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} z). && \text{by } (\text{Mem})' \end{aligned} \tag{LD}$$

□

## A.3 A proof of Theorem 6.1

**Theorem 6.1.** The four EqSSCL-axioms (Or), (Abs), (Mem), and (Comm) imply idempotence and associativity of $\curlywedge$ and $\curlyvee$, the double negation shift $\neg\neg x = x$ (that is, axiom (F3)), and the equations

$$(x \curlyvee \neg x) \curlywedge y = y, \tag{Tdef}$$
$$x \curlywedge (y \curlyvee z) = (x \curlywedge y) \curlyvee (x \curlywedge z). \tag{LD}$$

Furthermore, if $|A| \geq 2$, these four axioms are independent.

*Proof.* With help of the theorem prover *Prover9* [12]. Observe that $x \curlyvee y = y \curlyvee x$ readily follows from the axioms (Or) and (Comm); we refer to this equation by (Comm)$'$.

**Idempotence of $\curlywedge$.** We first derive

$$\begin{aligned}(y \curlywedge x) \curlywedge ((y \curlyvee z) \curlywedge x) &= (y \curlywedge x) \curlywedge ((y \curlywedge x) \curlyvee (\neg y \curlywedge (z \curlywedge x))) && \text{by (Mem), (Comm)}'\\ &= y \curlywedge x, && \text{by (Abs)}\end{aligned}$$

hence

$$(x \curlywedge y) \curlywedge ((y \curlyvee z) \curlywedge x) = y \curlywedge x. \qquad \text{by the above and (Comm)} \tag{43}$$

Finally,

$$\begin{aligned}x &= (x \curlyvee y) \curlywedge x && \text{by (Abs), (Comm)}\\ &= (x \curlywedge (x \curlyvee y)) \curlywedge (((x \curlyvee y) \curlyvee x) \curlywedge x) && \text{by (43) (substitute } x \curlyvee y \text{ for } y, \text{ and } x \text{ for } z)\\ &= x \curlywedge (((x \curlyvee y) \curlyvee x) \curlywedge x) && \text{by (Abs)}\\ &= x \curlywedge (x \curlywedge (x \curlyvee (x \curlyvee y))) && \text{by (Comm), (Comm)}'\\ &= x \curlywedge x. && \text{by (Abs)}\end{aligned}$$

**Idempotence of $\curlyvee$.** We first derive three auxiliary results:

$$\begin{aligned}x \curlyvee y &= (x \curlyvee y) \curlywedge (x \curlyvee y)\\ &= (x \curlywedge (x \curlyvee y)) \curlyvee (\neg x \curlywedge (y \curlywedge (x \curlyvee y))) && \text{by (Mem), (Comm)}'\\ &= x \curlyvee (\neg x \curlywedge (y \curlywedge (x \curlyvee y))) && \text{by (Abs)}\\ &= x \curlyvee (\neg x \curlywedge y), && \text{by (Comm)}', \text{ (Abs)} \qquad (44)\end{aligned}$$

$$\begin{aligned}x &= (x \curlyvee y) \curlywedge x && \text{by (Abs), (Comm)}\\ &= (\neg x \curlywedge (y \curlywedge x)) \curlyvee (x \curlywedge x) && \text{by (Mem)}\\ &= x \curlyvee (\neg x \curlywedge (y \curlywedge x)) && \text{by idempotence of } \curlywedge \text{ and (Comm)}'\\ &= x \curlyvee (y \curlywedge x) && \text{by (44)}\\ &= x \curlyvee (x \curlywedge y), && \text{by (Comm)} \qquad (45)\end{aligned}$$

$$\begin{aligned}(x \curlyvee y) \curlywedge (x \curlyvee z) &= (\neg x \curlywedge (y \curlywedge (x \curlyvee z))) \curlyvee (x \curlywedge (x \curlyvee z)) && \text{by (Mem)}\\ &= x \curlyvee (\neg x \curlywedge (y \curlywedge (x \curlyvee z))) && \text{by (Abs), (Comm)}'\\ &= x \curlyvee (y \curlywedge (x \curlyvee z)). && \text{by (44)} \qquad (46)\end{aligned}$$

Hence,

$$\begin{aligned}x \curlyvee x &= (x \curlyvee x) \curlywedge (x \curlyvee x)\\ &= x \curlyvee (x \curlywedge (x \curlyvee x)) && \text{by (46)}\\ &= x. && \text{by (45)}\end{aligned}$$

**Double negation shift.** With idempotence, the double negation shift follows immediately:

$$\neg\neg x = \neg(\neg x \mathbin{\curlywedge} \neg x) \overset{(\mathrm{Or})}{=} x \mathbin{\curlyvee} x = x.$$

Hence the duality principle holds in the setting without $\mathsf{T}$ and $\mathsf{F}$, which justifies the name (Comm)′ for the equation $x \mathbin{\curlyvee} y = y \mathbin{\curlyvee} x$.

**Equation** (Tdef), that is $(x \mathbin{\curlyvee} \neg x) \mathbin{\curlywedge} y = y$. First derive

$$\begin{aligned} x \mathbin{\curlywedge} (x \mathbin{\curlywedge} y) &= (x \mathbin{\curlywedge} y) \mathbin{\curlywedge} x && \text{by (Comm)} \\ &= (x \mathbin{\curlywedge} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) && \text{by (45)} \\ &= (x \mathbin{\curlywedge} y) \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlyvee} x) && \text{by (Comm)}' \\ &= x \mathbin{\curlywedge} y, && \text{by (Abs)} \end{aligned} \tag{47}$$

hence

$$\begin{aligned} (x \mathbin{\curlyvee} \neg x) \mathbin{\curlywedge} y &= (\neg x \mathbin{\curlywedge} (\neg x \mathbin{\curlywedge} y)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y) && \text{by (Mem)} \\ &= (\neg x \mathbin{\curlywedge} y) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y) && \text{by (47)} \\ &= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} y)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y) && \text{by idempotence} \\ &= (x \mathbin{\curlyvee} y) \mathbin{\curlywedge} y && \text{by (Mem)} \\ &= y. && \text{by (Comm), (Comm)′, (Abs)} \end{aligned} \tag{Tdef}$$

For the remaining statements of the theorem, that is, associativity and left-distributivity (LD), we can refer to the associated EqMSCL-derivations: by duality, equation (Tdef), and the observation that in each EqMSCL-derivation, the constant $\mathsf{T}$ can be represented by $u \mathbin{\curlyvee} \neg u$ with $u$ a fresh variable, the counterparts of the axioms (Tand) and (Neg) are available, and therefore the EqMSCL-derivations of these equations can be adapted in this way.

The independence of the four EqSSCL-axioms (Or), (Abs), (Mem), and (Comm) requires that $|A| \geq 2$ and is proved in Appendix A.4. □

## A.4 A proof of Theorem 6.3

**Theorem 6.3.** The axioms of EqSSCL are independent.

*Proof.* All independence models were found with the tool *Mace4* [12]. In each model $\mathbb{M}$ defined below, $[\![\mathsf{F}]\!]^{\mathbb{M}} = 0$ and $[\![\mathsf{T}]\!]^{\mathbb{M}} = 1$. Recall that $A \neq \emptyset$ and observe that one atom $a$ is used to show the independence of axioms (Or), (Mem), and (Comm). The independence result stated in Theorem 6.1 follows by using in the refutations below two atoms instead of $\mathsf{F}$ and $\mathsf{T}$, so this result requires that $|A| \geq 2$.

Independence of axiom (Neg). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Neg)$\}$ with domain $\{0,1,2,3\}$ that refutes $\mathsf{F} = \neg\mathsf{T}$ is the following:

| $\neg$ | |
|---|---|
| 0 | 3 |
| 1 | 2 |
| 2 | 1 |
| 3 | 0 |

| $\curlywedge$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 2 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 2 | 3 | 2 | 3 |

| $\curlyvee$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 2 | 3 |
| 3 | 1 | 1 | 3 | 3 |

Independence of axiom (Or). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Or)$\}$ with domain $\{0,1,2\}$ and $[\![a]\!]^{\mathbb{M}} = 2$ for some $a \in A$ that refutes $\mathsf{F} \curlyvee a = \neg(\neg\mathsf{F} \curlywedge \neg a)$ is the following:

| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | 0 |

| $\curlywedge$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 |
| 2 | 0 | 2 | 2 |

| $\curlyvee$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 1 | 2 |

Independence of axiom (Tand). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Tand)$\}$ with domain $\{0,1\}$ that refutes $\mathsf{T} \curlywedge \mathsf{F} = \mathsf{F}$ is the following:

| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |

| $\curlywedge$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| $\curlyvee$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Independence of axiom (Abs). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Abs)$\}$ with domain $\{0,1\}$ that refutes $\mathsf{T} \curlywedge (\mathsf{T} \curlyvee \mathsf{F}) = \mathsf{T}$ is the following:

| $\neg$ | |
|---|---|
| 0 | 0 |
| 1 | 0 |

| $\curlywedge$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| $\curlyvee$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |

Independence of axiom (Mem). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Mem)$\}$ with domain $\{0,1,2\}$ and $[\![a]\!]^{\mathbb{M}} = 2$ for some $a \in A$ that refutes $(\mathsf{F} \curlyvee \mathsf{F}) \curlywedge a = (\neg\mathsf{F} \curlywedge (\mathsf{F} \curlywedge a)) \curlyvee (\mathsf{F} \curlywedge a)$ is the following:

| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | 0 |

| $\curlywedge$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 1 | 0 | 1 | 2 |
| 2 | 2 | 2 | 0 |

| $\curlyvee$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 |

Independence of axiom (Comm). A model $\mathbb{M}$ for EqSSCL $\setminus \{$(Comm)$\}$ with domain $\{0,1,2\}$ and $[\![a]\!]^{\mathbb{M}} = 2$ for some $a \in A$ that refutes $\mathsf{F} \curlywedge a = a \curlywedge \mathsf{F}$ is the following:

| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | 2 |

| $\curlywedge$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 |
| 2 | 2 | 2 | 2 |

| $\curlyvee$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |

□