Non-commutative propositional logic with short-circuit evaluation

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

DOI
10.1080/11663081.2021.2010954
Publication date
2021
Document Version
Final published version
Published in
Journal of Applied Non-Classical Logics

JOURNAL OF APPLIED NON-CLASSICAL LOGICS
2021, VOL. 31, NOS. 3–4, 234–278
https://doi.org/10.1080/11663081.2021.2010954

Taylor & Francis
Taylor & Francis Group

Non-commutative propositional logic with short-circuit evaluation

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

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

ABSTRACT

Short-circuit evaluation denotes the semantics of propositional connectives in which the second argument is evaluated only if the first is insufficient to determine the value of the expression. Compound statements are evaluated from left to right. Short-circuit evaluation is widely used in programming, with negation and sequential conjunction and disjunction as primitives. We study the question of which laws axiomatise short-circuit evaluation. In MSCL (memorising short-circuit logic), atoms (propositional variables) evaluate to true or false, and in the evaluation of a compound statement, the first evaluation result of each atom is memorised. Hence, MSCL is 'Non-commutative propositional logic with short-circuit evaluation' and atomic evaluations cannot cause a side effect. Next, we consider the case that atoms can also evaluate to the truth value 'undefined'. For two- and three-valued MSCL, we present evaluation trees as an intuitive semantics and provide complete independent equational axiomatisations.

ARTICLE HISTORY
Received 28 October 2018
Accepted 29 September 2021

KEYWORDS
Non-commutative conjunction; conditional composition; sequential connectives; short-circuit evaluation; side effect

1. Introduction

This paper is about non-commutative propositional logic, which can be motivated by two simple examples from the setting of programming under the assumption that *short-circuit evaluation* is the evaluation strategy for the Boolean connectives: the second argument is evaluated only if the first argument does not suffice to determine the (evaluation) value of the expression.

(1) Consider the evaluation of a Boolean condition of the form

$$(f(x) \neq 0) \wedge (g(x,y) > 17)$$

for functions f and g, where it is known (or very likely) that the evaluation of the first conjunct is easier/faster than that of the second conjunct. This can be a reason to distinguish this condition from its symmetric counterpart.

CONTACT Alban Ponse a.ponse@uva.nl https://staff.fnwi.uva.nl/a.ponse

(2) Consider the evaluation of the more specific condition

$$(x \neq 0) \wedge (y/x > 17),$$

where the second conjunct can only be evaluated if the first one is *true*. This is a very good reason to distinguish this condition from its symmetric counterpart.

We consider sequential conjunction as the primary connective and sequential disjunction as a derived connective, and we use the asymmetric notations

$$\mathbin{\circledwedge} \quad \text{and} \quad \mathbin{\circledvee}$$

that emphasise their left-to-right interpretation: the small circle indicates that the left argument must be evaluated first (this notation stems from Bergstra et al. (1995)). Short-circuit evaluation combines well with negation, and sequential (equational) versions of De Morgan's laws are valid, such as

$$\neg(x \mathbin{\circledwedge} y) = \neg x \mathbin{\circledvee} \neg y.$$

In Section 8, we briefly discuss some other variants of sequential connectives.

A natural question is: *Which logical laws axiomatise short-circuit evaluation*? There are different answers to this question, depending on assumptions about the type of side effects that may occur and the different truth values used. Concerning the latter, we distinguish the following two cases:

(A) The case in which atoms (propositional variables) always evaluate to either *true* or *false*, thus the case of two-valued propositional logic with sequential connectives.
(B) The case that atoms always evaluate to one of the three truth values *true*, *false*, or *undefined* (as in the second example).

We first discuss case **(A)** and briefly recall *free short-circuit logic*, FSCL for short (Bergstra & Ponse, 2012; Bergstra et al., 2013; Ponse & Staudt, 2018). 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 the value of the expression. For example, in FSCL the sequential proposition $a \mathbin{\circledwedge} a$ is not equivalent with a or with $a \mathbin{\circledwedge} (a \mathbin{\circledvee} a)$. So FSCL is a logic for reasoning about sequential propositions that may have atomic side effects without any restriction, or, stated differently, a logic that is immune to atomic side effects. In Ponse and Staudt (2018) an equational axiomatisation of FSCL is provided, three typical axioms are $(x \mathbin{\circledwedge} y) \mathbin{\circledwedge} z = x \mathbin{\circledwedge} (y \mathbin{\circledwedge} z)$, $\mathsf{F} \mathbin{\circledwedge} x = \mathsf{F}$, and $x \mathbin{\circledwedge} \mathsf{F} = \neg x \mathbin{\circledwedge} \mathsf{F}$ (where the constant F represents falsehood).

The logic MSCL, *memorising short-circuit logic* (Bergstra & Ponse, 2012; Bergstra et al., 2013), is meant for 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 memorised. In this logic the sequential connectives are not commutative, for example, $a \mathbin{\circledwedge} \mathsf{F}$ and $\mathsf{F} \mathbin{\circledwedge} a$ are not equivalent (the first sequential proposition requires evaluation of atom a, the second does not). We provide a complete equational axiomatisation of MSCL. Some typical MSCL -consequences are $x \mathbin{\circledwedge} x = x$ and $x \mathbin{\circledwedge} (y \mathbin{\circledwedge} x) = x \mathbin{\circledwedge} y$. Also, each FSCL -axiom holds in MSCL (MSCL comprises FSCL).

The logic SSCL, *static short-circuit logic* (Bergstra & Ponse, 2012; Bergstra et al., 2013), is for reasoning about sequential propositions with the property that atomic side effects do not occur *and* that the sequential connectives are commutative. Typical laws for SSCL are those of equational propositional logic, and short-circuit evaluation is nothing more than a possible evaluation strategy. Of course, SSCL comprises MSCL.

Case (B). In this case, the starting point is that atoms, and therefore sequential propositions, can evaluate not only to truth and falsehood, but also to a third truth value *undefined*. We use the constant U to represent undefinedness. Three typical laws we adopt come from McCarthy's three-valued logic (McCarthy, 1963):

$$\mathsf{U} \mathbin{\circledwedge} x = \mathsf{U} \quad \text{and} \quad \neg\mathsf{U} = \mathsf{U} \quad \text{and} \quad \mathsf{F} \mathbin{\circledwedge} x = \mathsf{F}.$$

We provide a definition of $\mathrm{FSCL}^{\mathsf{U}}$, *free short-circuit logic with undefinedness*, which is an extension of FSCL with the constant U that implies the above laws. We present a set of equational axioms for $\mathrm{FSCL}^{\mathsf{U}}$. However, we do not have a completeness result and leave this as an open question.

Next, we define $\mathrm{MSCL}^{\mathsf{U}}$, *memorising short-circuit logic with undefinedness*, as an extension of MSCL with the constant U that also implies the three laws mentioned above. Extending the equational axiomatisation for MSCL with the axiom $\neg\mathsf{U} = \mathsf{U}$ provides a complete axiomatisation of $\mathrm{MSCL}^{\mathsf{U}}$. Also, $\mathrm{MSCL}^{\mathsf{U}}$ comprises $\mathrm{FSCL}^{\mathsf{U}}$.

The two-valued logic SSCL cannot be extended with the constant U and these three laws because commutativity of $\mathbin{\circledwedge}$ implies $\mathsf{F} = \mathsf{F} \mathbin{\circledwedge} \mathsf{U} = \mathsf{U} \mathbin{\circledwedge} \mathsf{F} = \mathsf{U}$.

Structure of the paper and main results. In Section 2, we discuss evaluation trees, which model the evaluation of a sequential proposition and were introduced in Staudt (2012). We recall the main results on FSCL, in particular, its equational axiomatisation EqFSCL for closed terms.

In Section 3, we define 'memorising evaluation trees' by a transformation on the evaluation trees for FSCL, and memorising *se*-congruence as the equality of such trees.

In Section 4, we provide an (equational) axiomatisation EqMSCL of memorising *se*-congruence. Furthermore, we show that the axioms of EqFSCL are derivable from EqMSCL, and discuss some consequences and an alternative axiomatisation.

In Section 5, we recall the definitions of the short-circuit logics mentioned above. These definitions stem from Bergstra et al. (2013), Bergstra and Ponse (2012) and employ the *conditional* – a ternary connective introduced by Hoare (1985) – as a hidden operator.

In Section 6, we prove that EqMSCL corresponds with MSCL in the sense that both define the same equational theory, so EqMSCL is an equational axiomatisation of MSCL.

In Section 7, we consider the constant U and adjust the definition of evaluation trees accordingly. We prove that $\mathrm{EqMSCL}^{\mathsf{U}} = \mathrm{EqMSCL} \cup \{\neg\mathsf{U} = \mathsf{U}\}$ is a complete axiomatisation of memorising *se*-congruence with U. Then, we provide definitions of $\mathrm{FSCL}^{\mathsf{U}}$ and $\mathrm{MSCL}^{\mathsf{U}}$ and prove that $\mathrm{EqMSCL}^{\mathsf{U}}$ axiomatises $\mathrm{MSCL}^{\mathsf{U}}$.

In Section 8, we discuss some issues related to three-valued logics. Then we provide an alternative definition of SSCL that is based on four simple axioms for Hoare's conditional connective.

In Section 9, we come up with some conclusions and open questions.

Notes. (1) All derivations from equational axiomatisations were checked with the theorem prover *Prover9*, and all independence results were found with help of the tool *Mace4*, see McCune (2008) for both tools. Detailed yet readable summaries of these proofs are added as appendices.

(2) Combined with Ponse and Staudt (2018), this paper subsumes most of Bergstra et al. (2013). Two topics discussed in the last paper are 'repetition-proof' and 'contractive' short-circuit logic; we aim to cover these topics in a future paper.

2. Evaluation trees and axioms for short-circuit evaluation

In this section, we summarise the main results of Ponse and Staudt (2018): evaluation trees and an axiomatisation of their equality are presented.

Given a non-empty, countable 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

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

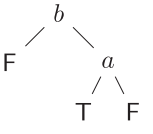
The operator $\_ \trianglelefteq a \trianglerighteq \_$ is called *tree composition over* a. In the evaluation tree $X \trianglelefteq a \trianglerighteq Y$, the root is represented by a, the left branch by X and the right branch by Y.

The *depth* $d : \mathcal{T}_A \to \mathbb{N}$ of an evaluation tree is defined by $d(\mathsf{T}) = d(\mathsf{F}) = 0$ and $d(Y \trianglelefteq a \trianglerighteq Z) = 1 + \max(d(Y), d(Z))$.

The leaves of an evaluation tree represent evaluation results (so we use the constants T and 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 T with Y and 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

$$\begin{aligned}
&X[\mathsf{T} \mapsto Y_1,\ \mathsf{F} \mapsto Z_1][\mathsf{T} \mapsto Y_2,\ \mathsf{F} \mapsto Z_2]\\
&\quad = X[\mathsf{T} \mapsto Y_1[\mathsf{T} \mapsto Y_2,\ \mathsf{F} \mapsto Z_2],\ \mathsf{F} \mapsto Z_1[\mathsf{T} \mapsto Y_2,\ \mathsf{F} \mapsto Z_2]].
\end{aligned}$$

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

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

where T is a constant for the truth value *true* and F for *false*, and refer to its signature by

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

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

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

$$\begin{aligned}
se(\mathsf{T}) &= \mathsf{T}, \quad se(\neg P) = se(P)[\mathsf{T} \mapsto \mathsf{F},\ \mathsf{F} \mapsto \mathsf{T}],\\
se(\mathsf{F}) &= \mathsf{F}, \quad se(P \mathbin{\diamond\!\wedge} Q) = se(P)[\mathsf{T} \mapsto se(Q)],\\
se(a) &= \mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F}, \quad se(P \mathbin{\diamond\!\vee} Q) = se(P)[\mathsf{F} \mapsto se(Q)].
\end{aligned}$$

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

$$\begin{aligned}
se(\neg b \mathbin{\diamond\!\wedge} a) &= se(\neg b)[\mathsf{T} \mapsto se(a)] = (\mathsf{F} \trianglelefteq b \trianglerighteq \mathsf{T})[\mathsf{T} \mapsto se(a)]\\
&= \mathsf{F} \trianglelefteq b \trianglerighteq (\mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F}),
\end{aligned}$$

which can be visualised as in (Picture 1). Also, $se(\neg(b \mathbin{\diamond\!\vee} \neg a)) = \mathsf{F} \trianglelefteq b \trianglerighteq (\mathsf{T} \trianglelefteq a \trianglerighteq \mathsf{F})$. An evaluation tree $se(P)$ represents short-circuit evaluation in a way that can be compared to a truth table for propositional logic, in that every possible evaluation of P is represented. However, there are some important differences with truth tables: in

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

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

se(*P*), the sequentiality of *P*'s evaluation is represented, and the same atom can 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 Staudt (2012), Ponse and Staudt (2018), it is proved that the axioms in Table 1 constitute an equational axiomatisation of *se*-congruence:[1]

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

This implies that the axioms in Table 1 axiomatise free short-circuit logic FSCL (defined in Section 5) for closed terms, and for this reason, this set of axioms is named EqFSCL. Some comments: axioms (F1)–(F3) imply sequential versions of De Morgan's laws, and thus a sequential variant of the duality principle. Axioms (F4)–(F6) define how the constants T and F interact with the sequential connectives, and axiom (F7) defines the associativity of $\curlywedge$.

Axiom (F8) defines a typical property of a logic that characterises immunity for side effects: not only are the evaluation results of $x \curlywedge \mathsf{F}$ and $\neg x \curlywedge \mathsf{F}$ always *false*, but for any $P \in \mathcal{S}_A$ the evaluations of *P* and $\neg P$ accumulate the same side effects.

Axiom (F9) expresses another property that concerns possible side effects: because the evaluation result of $x \curlywedge \mathsf{F}$ is always *false*, *y* is always evaluated in $(x \curlywedge \mathsf{F}) \curlyvee y$ and determines the evaluation result, which is also the case in $(x \curlyvee \mathsf{T}) \curlywedge y$. Note that the evaluations of $P \curlyvee \mathsf{T}$ and $P \curlywedge \mathsf{F}$ accumulate the same side effects.

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

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

$$\mathsf{T}^{dl} = \mathsf{F}, \quad a^{dl} = a, \quad (P \curlywedge Q)^{dl} = P^{dl} \curlyvee Q^{dl},$$
$$\mathsf{F}^{dl} = \mathsf{T}, \quad (\neg P)^{dl} = \neg P^{dl}, \quad (P \curlyvee Q)^{dl} = P^{dl} \curlywedge Q^{dl}.$$

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 $(x \mathbin{\diamond\!\vee} y) \mathbin{\diamond\!\vee} z = x \mathbin{\diamond\!\vee} (y \mathbin{\diamond\!\vee} z)$. From (F1)–(F3) it easily follows that EqFSCL satisfies the duality principle, that is, for all terms s, t over $\Sigma_{\mathrm{SCL}}(A)$,

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

We conclude this section with two properties of EqFSCL that were proved in Ponse and Staudt (2018).

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

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

and the axioms of EqFSCL^- are independent if A contains at least two atoms.

We kept (F1) and (F3) in EqFSCL because omitting them hinders intuition too much.

3. Evaluation trees for memorising short-circuit evaluation

In this section, we introduce memorising evaluation trees and 'memorising *se*-congruence'.

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

$$x \mathbin{\diamond\!\wedge} (x \mathbin{\diamond\!\vee} y) = x. \tag{Abs}$$

Equation (Abs) can be explained as follows: if x evaluates to *false*, then $x \mathbin{\diamond\!\wedge} (x \mathbin{\diamond\!\vee} y)$ evaluates to *false* as a result of the evaluation of the left occurrence of x (and $(x \mathbin{\diamond\!\vee} y)$ is not evaluated); if x evaluates to *true*, the second evaluation of x in the subterm $(x \mathbin{\diamond\!\vee} y)$ also results in *true* (because it is memorising) and therefore y is not evaluated.

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

$$(x \mathbin{\diamond\!\vee} y) \mathbin{\diamond\!\wedge} z = (\neg x \mathbin{\diamond\!\wedge} (y \mathbin{\diamond\!\wedge} z)) \mathbin{\diamond\!\vee} (x \mathbin{\diamond\!\wedge} z). \tag{Mem}$$

If x evaluates to *true*, then z determines the evaluation result of both expressions because the evaluation result of x is memorised; if x evaluates to *false*, the evaluation result of both expressions is determined by $y \mathbin{\diamond\!\wedge} z$ because the right disjunct $(x \mathbin{\diamond\!\wedge} z)$ of the right-hand side also evaluates to *false*.

Below we define the *memorising evaluation function* as a transformation on evaluation trees. This transformation implements the characteristic of memorising 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, memorising evaluations are those of propositional logic, except that the sequential connectives are not commutative. For example, $a \mathbin{\diamond\!\wedge} b$ and $b \mathbin{\diamond\!\wedge} a$ represent different evaluations, and thus are not equivalent.

Definition 3.1: The unary *memorising evaluation function*

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

yields *memorising 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 ($a \in A$):

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

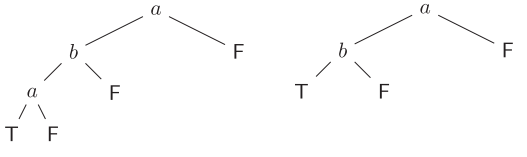
For $a \in A$, the auxiliary functions $L_a : \mathcal{T}_A \to \mathcal{T}_A$ ('Left a-reduction') and $R_a : \mathcal{T}_A \to \mathcal{T}_A$ ('Right a-reduction') are defined by

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

and

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

As an example we depict $se(a \mathbin{\diamond\!\wedge} (b \mathbin{\diamond\!\wedge} a))$ and the memorising evaluation tree $mse(a \mathbin{\diamond\!\wedge} (b \mathbin{\diamond\!\wedge} a))$:



From a more general point of view, a memorising evaluation tree is a *decision tree*, that is, a labelled, rooted, binary tree with internal nodes labelled from A and leaves labelled from $\{\mathsf{T}, \mathsf{F}\}$ such that for any path from the root to a leaf, the internal nodes receive distinct labels (cf. Moret, 1982). We note that the number of semantically different formulas is bounded by a function on $|A|$, which is an essential difference with the short-circuit logic FSCL. For $|A| = n$ (recall $n > 0$), the number of memorising evaluation trees is $T_n = n(T_{n-1})^2 + 2$ with $T_0 = 2$ (so the first few values are 6, 74, 16430).[2]

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

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

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

In the remainder of this section, we prove that $=_{mse}$ is indeed a congruence, and we start with some auxiliary results.

Lemma 3.3: *For all $a, b \in A$, $f, f' \in \{L, R\}$, and $X, Y \in \mathcal{T}_A$,*

(1) *If $b \neq a$ then $f_a(f'_b(X)) = f'_b(f_a(X))$,*
(2) $f_a(m(f_a(X))) = f_a(m(X))$,
(3) $f_a(m(X)) = m(f_a(X))$,
(4) $m(f_a(X[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}])) = m(f_a(X))[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}]$,
(5) $f_a(X[\mathsf{T} \mapsto Y]) = f_a(X)[\mathsf{T} \mapsto f_a(\mathsf{Y})]$,
(6) *If $m(X) = m(Y)$ then $m(f_a(X)) = m(f_a(Y))$,*
(7) $m(f_a(m(X))) = m(f_a(X))$.

Proof: See Appendix A.1. ■

Definition 3.4: On $\mathcal{T}_A$, define the auxiliary functions

$$\begin{aligned} \widetilde{\neg} X &= X[\mathsf{T} \mapsto \mathsf{F},\ \mathsf{F} \mapsto \mathsf{T}], \\ X \mathbin{\widetilde{\wedge}} Y &= X[\mathsf{T} \mapsto Y], \\ X \mathbin{\widetilde{\vee}} Y &= X[\mathsf{F} \mapsto Y]. \end{aligned}$$

Hence, $\widetilde{\neg}(se(P)) = se(\neg P)$, $se(P) \mathbin{\widetilde{\wedge}} se(Q) = se(P \wedge Q)$, and $se(P) \mathbin{\widetilde{\vee}} se(Q) = se(P \vee Q)$.

Lemma 3.5: *The relation $=_{mse}$ is a congruence on $\mathcal{S}_A$.*

Proof: It suffices to show that for $X, Y \in \mathcal{T}_A$, if $m(X) = m(X')$ and $m(Y) = m(Y')$, then $m(\widetilde{\neg} X) = m(\widetilde{\neg} X')$, $m(X \mathbin{\widetilde{\wedge}} Y) = m(X' \mathbin{\widetilde{\wedge}} Y')$, and $m(X \mathbin{\widetilde{\vee}} Y) = m(X' \mathbin{\widetilde{\vee}} Y')$.

The case for $m(\widetilde{\neg} X$ follows by case distinction on the form of X. The base cases $X \in \{\mathsf{T}, \mathsf{F}\}$ are trivial, and if $X = X_1 \trianglelefteq a \trianglerighteq X_2$, then $m(X) = m(X')$ implies that $X' = X'_1 \trianglelefteq a \trianglerighteq X'_2$ for some $X'_1, X'_2 \in \mathcal{T}_A$, and

$$m(L_a(X_1)) = m(L_a(X'_1)) \quad \text{and} \quad m(R_a(X_2)) = m(R_a(X'_2)). \tag{Aux1}$$

Write [neg] for the leaf replacement $[\mathsf{T} \mapsto \mathsf{F},\ \mathsf{F} \mapsto \mathsf{T}]$ and derive

$$\begin{aligned} m(\widetilde{\neg} X) &= m((X_1 \trianglelefteq a \trianglerighteq X_2)[\text{neg}]) \\ &= m(L_a(X_1[\text{neg}])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\text{neg}])) \\ &= m(L_a(X'_1[\text{neg}])) \trianglelefteq a \trianglerighteq m(R_a(X'_2[\text{neg}])) \quad \text{by La.3.3.4 and (Aux1)} \\ &= m(\widetilde{\neg} X'). \end{aligned}$$

The case for $m(X \mathbin{\widetilde{\wedge}} Y) = m(X' \mathbin{\widetilde{\wedge}} Y')$: for readability we split the proof obligation into two parts.

(A) $m(X \mathbin{\tilde{\barwedge}} Y) = m(X' \mathbin{\tilde{\barwedge}} Y)$. This follows by induction on the depth of X. The base cases $X \in \{\mathsf{T}, \mathsf{F}\}$ are simple (note that if $X = \mathsf{T}$ and $m(X) = m(X')$, then $X' = \mathsf{T}$ and we are done).

If $X = X_1 \trianglelefteq a \trianglerighteq X_2$ and $m(X) = m(X')$, then it must be the case that $X' = X'_1 \trianglelefteq a \trianglerighteq X'_2$ for some $X'_1, X'_2 \in \mathcal{T}_A$, and $m(L_a(X_1)) = m(L_a(X'_1))$ and $m(R_a(X_2)) = m(R_a(X'_2))$, thus (Aux1) holds. Derive

$$\begin{aligned}
m(X \mathbin{\tilde{\barwedge}} Y) &= m((X_1 \trianglelefteq a \trianglerighteq X_2) \mathbin{\tilde{\barwedge}} Y) \\
&= m(X_1[\mathsf{T} \mapsto Y] \trianglelefteq a \trianglerighteq X_2[\mathsf{T} \mapsto Y]) \\
&= m(L_a(X_1[\mathsf{T} \mapsto Y])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\mathsf{T} \mapsto Y])) \\
&= m(L_a(X_1)[\mathsf{T} \mapsto L_a(Y)]) \trianglelefteq a \trianglerighteq m(R_a(X_2)[\mathsf{T} \mapsto R_a(Y)]) \quad \text{by La.3.3.5} \\
&= m(L_a(X_1) \mathbin{\tilde{\barwedge}} L_a(Y)) \trianglelefteq a \trianglerighteq m(R_a(X_2) \mathbin{\tilde{\barwedge}} R_a(Y)) \\
&= m(L_a(X'_1) \mathbin{\tilde{\barwedge}} L_a(Y)) \trianglelefteq a \trianglerighteq m(R_a(X'_2) \mathbin{\tilde{\barwedge}} R_a(Y)) \quad \text{by IH and (Aux1)} \\
&= \cdots = m(X' \mathbin{\tilde{\barwedge}} Y).
\end{aligned}$$

(B) $m(X \mathbin{\tilde{\barwedge}} Y) = m(X \mathbin{\tilde{\barwedge}} Y')$. This follows by induction on the depth of X. The base cases $X \in \{\mathsf{T}, \mathsf{F}\}$ are trivial. If $X = X_1 \trianglelefteq a \trianglerighteq X_2$ derive

$$\begin{aligned}
m(X \mathbin{\tilde{\barwedge}} Y) &= m((X_1 \trianglelefteq a \trianglerighteq X_2) \mathbin{\tilde{\barwedge}} Y) \\
&= m(X_1[\mathsf{T} \mapsto Y] \trianglelefteq a \trianglerighteq X_2[\mathsf{T} \mapsto Y]) \\
&= m(L_a(X_1[\mathsf{T} \mapsto Y])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\mathsf{T} \mapsto Y])) \\
&= m(L_a(X_1)[\mathsf{T} \mapsto L_a(Y)]) \trianglelefteq a \trianglerighteq m(R_a(X_2)[\mathsf{T} \mapsto R_a(Y)]) \quad \text{by La.3.3.5} \\
&= m(L_a(X_1) \mathbin{\tilde{\barwedge}} L_a(Y)) \trianglelefteq a \trianglerighteq m(R_a(X_2) \mathbin{\tilde{\barwedge}} R_a(Y)) \\
&= m(L_a(X_1) \mathbin{\tilde{\barwedge}} L_a(Y')) \trianglelefteq a \trianglerighteq m(R_a(X_2) \mathbin{\tilde{\barwedge}} R_a(Y')) \quad \text{by La.3.3.6 and IH} \\
&= \cdots = m(X \mathbin{\tilde{\barwedge}} Y').
\end{aligned}$$

The proof for the case $m(X \mathbin{\tilde{\barvee}} Y) = m(X' \mathbin{\tilde{\barvee}} Y')$ is similar. ■

4. Axioms for memorising *se*-congruence

In this section, we provide an axiomatisation of memorising *se*-congruence. Furthermore, we provide a number of convenient consequences of this axiomatisation and briefly discuss an alternative axiomatisation.

In Table 2, we present a set of equational axioms for $=_{mse}$ and we call this set EqMSCL (this is a simplified version of EqMSCL as introduced in Bergstra & Ponse, 2012; Bergstra et al., 2013). To enhance readability, we renamed the EqFSCL-axioms used: (F1) → (Neg), (F2) → (Or), and (F4) → (Tand).

Theorem 4.1: *For all* $P, Q \in \mathcal{S}_A$, $\mathrm{EqMSCL} \vdash P = Q \Rightarrow P =_{mse} Q$.

Proof: By Lemma 3.5, the relation $=_{mse}$ is a congruence on $\mathcal{S}_A$, so it suffices to show that all closed instances of the EqMSCL-axioms satisfy $=_{mse}$. We first prove this for

Table 2. EqMSCL, a set of axioms for memorising *se*-congruence.

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

axiom (Abs). It suffices to show that for all $X, Y \in \mathcal{T}_A$, $m(X \mathbin{\widetilde{\curlywedge}} (X \mathbin{\widetilde{\curlyvee}} Y)) = m(X)$, which follows by structural induction on X. The base cases are trivial. If $X = X_1 \trianglelefteq a \trianglerighteq X_2$ we use a simple consequence of Lemma 3.3.5:

$$\text{For all } a \in A \text{ and } X, Y, Z \in \mathcal{T}_A,\ L_a(X[\mathsf{T} \mapsto (Y \trianglelefteq a \trianglerighteq Z)]) = L_a(X[\mathsf{T} \mapsto Y]) \text{ and } R_a(X[\mathsf{T} \mapsto (Y \trianglelefteq a \trianglerighteq Z)) = R_a(X[\mathsf{T} \mapsto Z]). \tag{Aux2}$$

Derive

$$\begin{aligned}
&m((X_1 \trianglelefteq a \trianglerighteq X_2) \mathbin{\widetilde{\curlywedge}} ((X_1 \trianglelefteq a \trianglerighteq X_2) \mathbin{\widetilde{\curlyvee}} Y)) \\
&= m(X_1[\mathsf{T} \mapsto B] \trianglelefteq a \trianglerighteq X_2[\mathsf{T} \mapsto B]) \quad \text{with } B = (X_1 \trianglelefteq a \trianglerighteq X_2)[\mathsf{F} \mapsto Y] \\
&= m(L_a(X_1[\mathsf{T} \mapsto B])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\mathsf{T} \mapsto B])) \\
&= m(L_a(X_1[\mathsf{T} \mapsto (X_1[\mathsf{F} \mapsto Y])])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\mathsf{T} \mapsto (X_2[\mathsf{F} \mapsto Y])])) \quad \text{by (Aux2)} \\
&= m(L_a(m(X_1[\mathsf{T} \mapsto (X_1[\mathsf{F} \mapsto Y])]))) \trianglelefteq a \trianglerighteq \\
&\qquad m(R_a(m(X_2[\mathsf{T} \mapsto (X_2[\mathsf{F} \mapsto Y])]))) \qquad \text{by La.3.3.7} \\
&= m(L_a(m(X_1 \mathbin{\widetilde{\curlywedge}} (X_1 \mathbin{\widetilde{\curlyvee}} Y)))) \trianglelefteq a \trianglerighteq m(R_a(m(X_2 \mathbin{\widetilde{\curlywedge}} (X_2 \mathbin{\widetilde{\curlyvee}} Y)))) \\
&= m(L_a(m(X_1))) \trianglelefteq a \trianglerighteq m(R_a(m(X_2))) \qquad \text{by IH} \\
&= m(L_a(X_1)) \trianglelefteq a \trianglerighteq m(R_a(X_2)) \qquad \text{by La.3.3.7} \\
&= m(X_1 \trianglelefteq a \trianglerighteq X_2).
\end{aligned}$$

In a similar way, it easily follows that all closed instances of the first three axioms (Neg), (Or), and (Tand) satisfy $=_{mse}$. The case for axiom (Mem) is a bit more complex, in Appendix A.2, we provide a detailed proof. ■

Before we prove that EqMSCL is a complete axiomatisation of memorising *se*-congruence (Theorem 4.9), we discuss some consequences of EqMSCL. The next theorem implies that *se*-congruence is subsumed by memorising *se*-congruence.

Theorem 4.2: EqMSCL $\vdash$ EqFSCL.

Proof: With help of the theorem prover *Prover9*, see Appendix A.3. ■

In the proof of Theorem 4.2, the EqFSCL-axioms are derived in a particular order to obtain useful intermediate results. The double negation shift $\neg\neg x = x$ (F3) is derived first, which justifies the use of the duality principle in subsequent derivations. For easy reference, we mention here some auxiliary results that are used in Section 6.

Fact 4.3: The following equations are derivable from EqMSCL (see Appendix A.3):

$$x \overset{(\mathrm{Abs})}{=} x \mathbin{\wedge\!\!\!\circ} (x \mathbin{\vee\!\!\!\circ} \mathsf{F}) \overset{(\mathrm{F5})}{=} x \mathbin{\wedge\!\!\!\circ} x, \tag{Idempotence}$$

$$x \mathbin{\wedge\!\!\!\circ} y = (x \mathbin{\wedge\!\!\!\circ} \mathsf{F}) \mathbin{\vee\!\!\!\circ} (x \mathbin{\wedge\!\!\!\circ} y), \tag{Ar1}$$

$$x \mathbin{\vee\!\!\!\circ} y = (\neg x \mathbin{\wedge\!\!\!\circ} y) \mathbin{\vee\!\!\!\circ} x. \tag{Ar2}$$

We note that $x \mathbin{\vee\!\!\!\circ} \mathsf{F} \overset{(\mathrm{F5})}{=} x$ easily follows from EqMSCL, and so do the auxiliary results (Ar1) and (Ar2).

Also, the following two equations are derivable from EqMSCL:

$$(x \mathbin{\wedge\!\!\!\circ} y) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} z) = (\neg x \mathbin{\vee\!\!\!\circ} y) \mathbin{\wedge\!\!\!\circ} (x \mathbin{\vee\!\!\!\circ} z), \tag{M1}$$

$$(x \mathbin{\wedge\!\!\!\circ} y) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} z) = (\neg x \mathbin{\wedge\!\!\!\circ} z) \mathbin{\vee\!\!\!\circ} (x \mathbin{\wedge\!\!\!\circ} y). \tag{M2}$$

With memorising evaluations, these terms express '`if x then y else z`'.

A typical EqMSCL-consequence is $x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} x) = x \mathbin{\wedge\!\!\!\circ} y$ (cf. the last example on memorising evaluation trees). First derive

$$\neg x \mathbin{\wedge\!\!\!\circ} \mathsf{F} \overset{(\mathrm{F8})}{=} x \mathbin{\wedge\!\!\!\circ} \mathsf{F} \overset{(\mathrm{Ar2})'}{=} (\neg x \mathbin{\vee\!\!\!\circ} \mathsf{F}) \mathbin{\wedge\!\!\!\circ} x \overset{(\mathrm{F5})}{=} \neg x \mathbin{\wedge\!\!\!\circ} x, \tag{C1}$$

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

$$\begin{aligned} x \mathbin{\wedge\!\!\!\circ} y &= (\neg x \mathbin{\vee\!\!\!\circ} y) \mathbin{\wedge\!\!\!\circ} x \quad \text{by (Ar2)}' \\ &= (x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} x)) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} x) \quad \text{by (Mem), (F3)} \\ &= (x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} x)) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} \mathsf{F}) \quad \text{by (C1)} \\ &= (\neg x \mathbin{\wedge\!\!\!\circ} \mathsf{F}) \mathbin{\vee\!\!\!\circ} (x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} x)) \quad \text{by (M2)} \\ &= x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} x). \quad \text{by (F8), (Ar1)} \end{aligned} \tag{C2}$$

Another convenient result on EqMSCL, used in Section 6, is the following.

Theorem 4.4: *The following equations are derivable from* EqMSCL*, where* (LD) *abbreviates left-distributivity of* $\mathbin{\wedge\!\!\!\circ}$*:*

$$x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\vee\!\!\!\circ} z) = (x \mathbin{\wedge\!\!\!\circ} y) \mathbin{\vee\!\!\!\circ} (x \mathbin{\wedge\!\!\!\circ} z), \tag{LD}$$

$$((x \mathbin{\wedge\!\!\!\circ} y) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} z)) \mathbin{\wedge\!\!\!\circ} u = (x \mathbin{\wedge\!\!\!\circ} (y \mathbin{\wedge\!\!\!\circ} u)) \mathbin{\vee\!\!\!\circ} (\neg x \mathbin{\wedge\!\!\!\circ} (z \mathbin{\wedge\!\!\!\circ} u)). \tag{M3}$$

Proof: With help of the theorem prover *Prover9*, see Appendix A.4. ■

In order to prove the completeness of EqMSCL, we use normal forms.

Definition 4.5: *Memorising* SCL *Normal Forms* (mSNFs) are inductively defined:

- T and F are mSNFs, and
- $(a \mathbin{\wedge\!\!\!\circ} P) \mathbin{\vee\!\!\!\circ} (\neg a \mathbin{\wedge\!\!\!\circ} Q)$ is an mSNF if $a \in A$, and P and Q are mSNFs that do not contain a.

We write MSNF for the set of all mSNFs.

The following functions on MSNF are used to compose mSNFs.

Definition 4.6: For $a \in A$, the function $\mathsf{T}_a : \mathrm{MSNF} \to \mathrm{MSNF}$ is defined by

$$\mathsf{T}_a(\mathsf{T}) = \mathsf{T}, \quad \mathsf{T}_a(\mathsf{F}) = \mathsf{F},$$

$$\mathsf{T}_a((b \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg b \mathbin{\owedge} P_2)) = \begin{cases} P_1 & \text{if } b = a, \\ (b \mathbin{\owedge} \mathsf{T}_a(P_1)) \mathbin{\overline{\vee}} (\neg b \mathbin{\owedge} \mathsf{T}_a(P_2)) & \text{otherwise.} \end{cases}$$

For $a \in A$, the function $\mathsf{F}_a : \mathrm{MSNF} \to \mathrm{MSNF}$ is defined by

$$\mathsf{F}_a(\mathsf{T}) = \mathsf{T}, \quad \mathsf{F}_a(\mathsf{F}) = \mathsf{F},$$

$$\mathsf{F}_a((b \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg b \mathbin{\owedge} P_2)) = \begin{cases} P_2 & \text{if } b = a, \\ (b \mathbin{\owedge} \mathsf{F}_a(P_1)) \mathbin{\overline{\vee}} (\neg b \mathbin{\owedge} \mathsf{F}_a(P_2)) & \text{otherwise.} \end{cases}$$

So, T_a removes the a-occurrences in an mSNF under the assumption that a evaluates to T (and F_a does this under the assumption that a evaluates to F). Note that for each $P \in \mathrm{MSNF}$, both $\mathsf{T}_a(P)$ and $\mathsf{F}_a(P)$ are also mSNFs. In order to compose mSNFs, we use the following lemma (for proof see Appendix A.5).

Lemma 4.7: *For all $a \in A$, $P \in \mathcal{S}_A$, and $Q \in \mathrm{MSNF}$,*

(1) $\mathrm{EqMSCL} \vdash a \mathbin{\owedge} (P \mathbin{\owedge} Q) = a \mathbin{\owedge} (P \mathbin{\owedge} \mathsf{T}_a(Q))$,
(2) $\mathrm{EqMSCL} \vdash \neg a \mathbin{\owedge} (P \mathbin{\owedge} Q) = \neg a \mathbin{\owedge} (P \mathbin{\owedge} \mathsf{F}_a(Q))$,
(3) $\mathrm{EqMSCL} \vdash a \mathbin{\owedge} (P \mathbin{\overline{\vee}} Q) = a \mathbin{\owedge} (P \mathbin{\overline{\vee}} \mathsf{T}_a(Q))$,
(4) $\mathrm{EqMSCL} \vdash \neg a \mathbin{\owedge} (P \mathbin{\overline{\vee}} Q) = \neg a \mathbin{\owedge} (P \mathbin{\overline{\vee}} \mathsf{F}_a(Q))$.

Lemma 4.8: *For each $P \in \mathcal{S}_A$ there is $P' \in \mathit{MSNF}$ such that* $\mathrm{EqMSCL} \vdash P = P'$.

Proof: We first prove three auxiliary results on mSNFs.

(1) For each mSNF P, there is $Q \in \mathrm{MSNF}$ such that $\mathrm{EqMSCL} \vdash \neg P = Q$. This follows by induction on the structure of P. If $P \in \{\mathsf{T}, \mathsf{F}\}$ this is trivial, and otherwise $P = (a \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} P_2)$. Now derive from EqMSCL

$$\neg((a \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} P_2)) = (a \mathbin{\owedge} \neg P_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} \neg P_2).$$

By induction, there are mSNFs Q_i such that $\mathrm{EqMSCL} \vdash \neg P_i = Q_i$ and that clearly do not contain a. Let $Q = (a \mathbin{\owedge} Q_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} Q_2)$, then $\mathrm{EqMSCL} \vdash \neg P = Q$.

(2) If $P, Q \in \mathrm{MSNF}$, then there is $R \in \mathrm{MSNF}$ such that $\mathrm{EqMSCL} \vdash P \mathbin{\owedge} Q = R$. This follows by induction on the structure of P. If $P \in \{\mathsf{T}, \mathsf{F}\}$ this is trivial, and otherwise $P = (a \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} P_2)$. Now apply Equation (M3):

$$((a \mathbin{\owedge} P_1) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} P_2)) \mathbin{\owedge} Q = (a \mathbin{\owedge} (P_1 \mathbin{\owedge} Q)) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} (P_2 \mathbin{\owedge} Q)).$$

By induction, there are mSNFs R_i such that $\mathrm{EqMSCL} \vdash P_i \mathbin{\owedge} Q = R_i$. Let $R = (a \mathbin{\owedge} \mathsf{T}_a(R_1)) \mathbin{\overline{\vee}} (\neg a \mathbin{\owedge} \mathsf{F}_a(R_2))$, then by Lemma 4.7.1-2, R is the *mSNF* that satisfies $\mathrm{EqMSCL} \vdash P \mathbin{\owedge} Q = R$.

(3) If $P, Q \in$ MSNF, then there is $R \in$ MSNF such that EqMSCL $\vdash P \veebar Q = R$. This also follows by induction on the structure of P. If $P \in \{\mathsf{T}, \mathsf{F}\}$ this is trivial, and otherwise, $P = (a \owedge P_1) \veebar (\neg a \owedge P_2)$. Now derive from EqMSCL

$$((a \owedge P_1) \veebar (\neg a \owedge P_2)) \veebar Q = (a \owedge (P_1 \veebar Q)) \veebar (\neg a \owedge (P_2 \veebar Q)).$$

By induction, there are mSNFs R_i such that EqMSCL $\vdash P_i \veebar Q = R_i$. Let $R = (a \owedge \mathsf{T}_a(R_1)) \veebar (\neg a \owedge \mathsf{F}_a(R_2))$, then by Lemma 4.7.3-4, EqMSCL $\vdash P \veebar Q = R$.

Next, we prove the lemma by induction on the structure of P. If $P \in \{\mathsf{T}, \mathsf{F}\}$ this follows immediately. If $P = a$, then

$$\begin{aligned}
\text{EqMSCL} \vdash a &= (a \owedge \mathsf{F}) \veebar a \quad \text{by idempotence and (Ar1)} \\
&= (\neg a \owedge \mathsf{F}) \veebar (a \owedge \mathsf{T}) \quad \text{by (F8) and (F5)}' \\
&= (a \owedge \mathsf{T}) \veebar (\neg a \owedge \mathsf{F}) \quad \text{by (M2).}
\end{aligned}$$

If $P = \neg Q_1$, $P = Q_1 \owedge Q_2$, or $P = Q_1 \veebar Q_2$, then by induction there are mSNFs R_i with EqMSCL $\vdash Q_i = R_i$. Now apply the appropriate auxiliary result. ■

Theorem 4.9: *For all* $P, Q \in \mathcal{S}_A$, EqMSCL $\vdash P = Q \iff P =_{mse} Q$.

Proof: ($\Rightarrow$) This is Theorem 4.1.

($\Leftarrow$) By Lemma 4.8 there are $P', Q' \in$ MSNF such that EqMSCL $\vdash P = P', Q = Q'$, and thus by Theorem 4.1, $P =_{mse} P'$ and $Q =_{mse} Q'$. Because $=_{mse}$ is a congruence, $P' =_{mse} Q'$. We show by structural induction on P' that $P' = Q'$, and thus EqMSCL $\vdash P = Q$. The base cases are trivial, and if $P' = (a \owedge P_1) \veebar (\neg a \owedge P_2)$, then it must be the case that $Q' = (a \owedge Q_1) \veebar (\neg a \owedge Q_2)$. Furthermore,

$$\begin{aligned}
mse(P') &= m(se((a \owedge P_1) \veebar (\neg a \owedge P_2))) \\
&= m(se(a \owedge P_1)[\mathsf{F} \mapsto se(\neg a \owedge P_2)]) \\
&= m(se(P_1) \trianglelefteq a \trianglerighteq se(\neg a \owedge P_2)) \\
&= m(L_a(se(P_1))) \trianglelefteq a \trianglerighteq m(R_a(se(\neg a \owedge P_2))) \\
&= m(se(P_1)) \trianglelefteq a \trianglerighteq m(R_a(\mathsf{F} \trianglelefteq a \trianglerighteq se(P_2))) \quad (a \text{ does not occur in } P_1) \\
&= mse(P_1) \trianglelefteq a \trianglerighteq mse(P_2), \quad (a \text{ does not occur in } P_2)
\end{aligned}$$

and $mse(Q') = mse(Q_1) \trianglelefteq a \trianglerighteq mse(Q_2)$ follows in a similar way. By induction, $P_i = Q_i$, and hence $P' = Q'$. ■

Theorem 4.10: *The axioms of* EqMSCL *are independent.*

Proof: By Theorem 7.12 (which states that a superset of EqMSCL is independent). ■

We end this section by mentioning two alternatives for EqMSCL.

Proposition 4.11: *Replacing axiom* (Mem) *in* EqMSCL *by* (M1)*, and either* (M3) *or*

$$((x \mathbin{\owedge} y) \mathbin{\overset{\circ}{\vee}} (\neg x \mathbin{\owedge} z)) \mathbin{\owedge} u = (\neg x \mathbin{\owedge} (z \mathbin{\owedge} u)) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\owedge} (y \mathbin{\owedge} u))$$

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

Both these sets of axioms are independent (by *Mace4*, McCune, 2008). With *Prover9* (McCune, 2008), derivations of (Ar1), (Ar2), (M2), and (Mem), respectively, are simple. Furthermore, associativity of $\owedge$ (F7) follows easily (in contrast to the proof of Theorem 4.2):

$$\begin{aligned}
(x \mathbin{\owedge} y) \mathbin{\owedge} z &= ((\neg x \mathbin{\owedge} \mathrm{F}) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\owedge} y)) \mathbin{\owedge} z \quad \text{by (Ar1), (F8)} \\
&= (\neg x \mathbin{\owedge} (\mathrm{F} \mathbin{\owedge} z)) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\owedge} (y \mathbin{\owedge} z)) \quad \text{by (M3), (F3)} \\
&= ((x \mathbin{\owedge} \mathrm{F}) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\owedge} (y \mathbin{\owedge} z))) \quad \text{by (F6), (F8)} \\
&= x \mathbin{\owedge} (y \mathbin{\owedge} z). \quad \text{by (Ar1)}
\end{aligned}$$

5. 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 $\Sigma_{\mathrm{SCL}}(A)$. Then, we recall the definitions of free short-circuit logic (FSCL), memorising short-circuit logic (MSCL), and static short-circuit logic (SSCL) that were published earlier.

In 1948, Church introduced (in Church, 1948) the *conditioned disjunction* connective $[p, q, r]$, which, following the author, may be read 'p or r according as q or not q'. Church showed that the conditioned disjunction together with constants t and f for truth en falsehood, form a complete set of independent primitive connectives for the propositional calculus. Finally, he also noted that for propositional variables p, q, r, the dual of $[p, q, r]$ is simply $[r, q, p]$, so that

> to dualize an expression of the propositional calculus in which the only connectives occurring are conditioned disjunction, t, and f, it is sufficient to write the expression backwards and at the same time to interchange the letters t and f.[3]

Apparently unaware of Church's conditioned disjunction, Hoare introduced the ternary *conditional* connective

$$p \lhd q \rhd r$$

(in Hoare, 1985) that has the same truth table as $[p, q, r]$ and provided eleven equational axioms to show that the conditional and two constants for truth and falsehood characterise the propositional calculus. We adhere to Hoare's notation.

A more common expression for the conditional $x \lhd y \rhd z$ is '`if` y `then` x `else` z', which emphasises 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 Hoare (1985), an equational axiomatisation of propositional logic is provided that only uses the conditional. Furthermore,

Table 3. The set CP of axioms for proposition algebra.

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

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 axiomatisation 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 8, we present a concise and simple alternative for this axiomatisation.[4]

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

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

The four axioms in Table 3, named CP (for Conditional Propositions), establish a complete axiomatisation 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 Bergstra and Ponse (2017, Thm.2.11) (and repeated in Ponse & Staudt, 2018, Appendix A3).

With the conditional connective and the constants T and F, the sequential connectives prescribing short-circuit evaluation are definable:

$$\neg x = \mathsf{F} \lhd x \rhd \mathsf{T}, \qquad \text{(defNeg)}$$

$$x \mathbin{\owedge} y = y \lhd x \rhd \mathsf{F}, \qquad \text{(defAnd)}$$

$$x \mathbin{\ovee} y = \mathsf{T} \lhd x \rhd y. \qquad \text{(defOr)}$$

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

$$\mathrm{CP}(\neg, \owedge, \ovee),$$

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

In order to capture memorising evaluations, the following axiom is formulated in Bergstra and Ponse (2011):

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

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

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

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 directly from CP_{mem}. Furthermore, if we replace u by 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 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* (McCune, 2008), it easily follows that the axioms of CP_{mem} are independent, and therefore, those of CP are also independent.

We write

$$\mathrm{CP}_{mem}(\neg, \owedge, \ovee)$$

for the axioms of CP_{mem} extended with Equations (defNeg)–(defOr). An important property of $\mathrm{CP}_{mem}(\neg, \owedge, \ovee)$ is that the conditional connective can be expressed with the sequential connectives and negation. First, observe that it is trivial to derive

$$\neg x \owedge z = \mathsf{F} \lhd x \rhd z, \tag{1}$$

and hence

$$\begin{aligned}
&(x \owedge y) \ovee (\neg x \owedge z)\\
&\quad = \mathsf{T} \lhd (y \lhd x \rhd \mathsf{F}) \rhd (\mathsf{F} \lhd x \rhd z) \quad \text{by (defNeg)–(defOr), (1)}\\
&\quad = (\mathsf{T} \lhd y \rhd (\mathsf{F} \lhd x \rhd z)) \lhd x \rhd (\mathsf{F} \lhd x \rhd z) \quad \text{by (CP4), (CP2)}\\
&\quad = (\mathsf{T} \lhd y \rhd \mathsf{F}) \lhd x \rhd z \quad \text{by (CPmem1), (CPcon1)}\\
&\quad = y \lhd x \rhd z. \quad \text{by (CP3)}
\end{aligned} \tag{2}$$

The following equation can be helpful, and can be similarly proved from $\mathrm{CP}_{mem}(\neg, \owedge, \overset{\circ}{\vee})$:

$$(x \overset{\circ}{\vee} z) \owedge (\neg x \overset{\circ}{\vee} y) = y \lhd x \rhd z. \tag{3}$$

In Bergstra and Ponse (2012) and Bergstra et al. (2013), a set-up is provided for defining (two-valued) short-circuit logics in a generic way with help of the conditional by restricting the consequences of some CP-axiomatisation extended with Equation (defNeg) (that is, $\neg x = \mathsf{F} \lhd x \rhd \mathsf{T}$) and Equation (defAnd) (i.e. $x \owedge y = y \lhd x \rhd \mathsf{F}$) 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* (Bergstra et al., 1990) 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 5.1: A *short-circuit logic* is a logic that implies the consequences of the module expression

$$\mathrm{SCL} = \{\mathsf{T}, \neg, \owedge\} \square (\mathrm{CP} \cup \{(\mathrm{defNeg}), (\mathrm{defAnd})\}).$$

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

$$\begin{aligned} \neg\neg x &= \mathsf{F} \lhd (\mathsf{F} \lhd x \rhd \mathsf{T}) \rhd \mathsf{T} \quad \text{by (defNeg)} \\ &= (\mathsf{F} \lhd \mathsf{F} \rhd \mathsf{T}) \lhd x \rhd (\mathsf{F} \lhd \mathsf{T} \rhd \mathsf{T}) \quad \text{by (CP4)} \\ &= \mathsf{T} \lhd x \rhd \mathsf{F} \quad \text{by (CP2), (CP1)} \\ &= x. \quad \text{by (CP3)} \end{aligned} \tag{4}$$

In Bergstra and Ponse (2012) and Bergstra et al. (2013), the following short-circuit logics are defined:

Definition 5.2: *Free short-circuit logic* (FSCL) is the short-circuit logic that implies no other consequences than those of the module expression SCL.

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

$$\{\mathsf{T}, \neg, \owedge\} \square (\mathrm{CP}_{mem} \cup \{(\mathrm{defNeg}), (\mathrm{defAnd})\}).$$

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

$$\{\mathsf{T}, \neg, \owedge\} \square (\mathrm{CP}_{mem} \cup \{(\mathrm{defNeg}), (\mathrm{defAnd}), \mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}\}).$$

To enhance readability, we extend these short-circuit logics with the constant F and its defining axiom (Neg), which is justified by the SCL-derivation

$$\begin{aligned} \mathsf{F} &= \mathsf{F} \lhd \mathsf{T} \rhd \mathsf{T} \quad \text{by (CP1)} \\ &= \neg \mathsf{T}, \quad \text{by (defNeg)} \end{aligned} \tag{5}$$

and with the connective $\barwedge$ and its defining axiom (Or) (thus, $x \barwedge y = \neg(\neg x \mathbin{\circledwedge} \neg y)$) by admitting equation (defOr) in SCL-derivations, that is, $x \barwedge y = \mathsf{T} \lhd x \rhd y$. This last extension is justified by

$$\begin{aligned}
\neg(\neg x \mathbin{\circledwedge} \neg y) &= \mathsf{F} \lhd (\neg y \lhd (\mathsf{F} \lhd x \rhd \mathsf{T}) \rhd \mathsf{F}) \rhd \mathsf{T} \quad \text{by (defNeg), (defAnd)} \\
&= \mathsf{F} \lhd (\mathsf{F} \lhd x \rhd \neg y) \rhd \mathsf{T} \quad \text{by (CP4), (CP2), (CP1)} \\
&= (\mathsf{F} \lhd \mathsf{F} \rhd \mathsf{T}) \lhd x \rhd (\mathsf{F} \lhd \neg y \rhd \mathsf{T}) \quad \text{by (CP4)} \\
&= \mathsf{T} \lhd x \rhd y. \quad \text{by (CP2), (defNeg), (4)}
\end{aligned} \tag{6}$$

In Staudt (2012), Ponse and Staudt (2018), the following results are proved:

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

In the remainder of the paper, we will prove a correspondence result for MSCL and EqMSCL, which establishes that the latter axiomatises MSCL.

6. Completeness of EqMSCL revisited

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

Given a signature Σ, we write

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

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

Definition 6.1: Define the following two functions between sets of open terms:

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

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

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

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

Lemma 6.2: *For all* $t \in \mathbb{T}_{\Sigma_{\mathsf{SCL}}(A),\mathcal{X}}$, $\mathsf{CP}_{mem}(\neg, \mathbin{\circledwedge}, \barwedge) \vdash f(t) = t$.

Proof: By structural induction on t. ■

Lemma 6.3: *For all* $s, t \in \mathbb{T}_{\Sigma_{\mathsf{CP}}(A),\mathcal{X}}$, $\mathsf{CP}_{mem}(\neg, \mathbin{\circledwedge}, \barwedge) \vdash s = t \Rightarrow \mathsf{CP}_{mem} \vdash s = t$.

Proof: In an equational proof of $\mathrm{CP}_{mem}(\neg, \owedge, \ovee) \vdash s = t$, each occurrence of one of the Equations (defNeg), (defAnd), and (defOr) 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 (defAnd) and (defOr).

Because s and t do not contain occurrences of $\neg, \owedge$, and $\ovee$, this yields an equational proof of $s = t$ in CP_{mem}. ■

Lemma 6.4: *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 CP_{mem}-axiom is derivable in EqMSCL.

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

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

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

The cases for axioms (CP4) and (CPmem) are spelled out in Appendix A.6. ■

Theorem 6.5: *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 (5).

Axiom (Or). This follows from (6).

Axiom (Tand). $\mathsf{T} \owedge x = x \lhd \mathsf{T} \rhd \mathsf{F} = x$.

Axiom (Abs). $x \owedge (x \ovee y) = (\mathsf{T} \lhd x \rhd y) \lhd x \rhd \mathsf{F} \overset{(\mathrm{CPcon2})}{=} \mathsf{T} \lhd x \rhd \mathsf{F} = x$.

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

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

($\Leftarrow$)

$$
\begin{aligned}
\mathrm{MSCL} \vdash s = t &\Rightarrow \mathrm{CP}_{mem}(\neg, \owedge, \ovee) \vdash s = t \quad \text{by definition} \\
&\Rightarrow \mathrm{CP}_{mem}(\neg, \owedge, \ovee) \vdash f(s) = f(t) \quad \text{by Lemma 6.2} \\
&\Rightarrow \mathrm{CP}_{mem} \vdash f(s) = f(t) \quad \text{by Lemma 6.3} \\
&\Rightarrow \mathrm{EqMSCL} \vdash g(f(s)) = g(f(t)). \quad \text{by Lemma 6.4}
\end{aligned}
$$

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

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

■

Theorems 4.9 and 6.5 lead to the following result.

Corollary 6.6: MSCL *axiomatises memorising se-congruence, and* EqMSCL *is an equational axiomatisation of* MSCL.

7. Short-circuit logics with undefinedness

In this section, we include the third truth value *undefined*, represented by the constant U. In the setting with the conditional connective, U is defined by the axiom

$$x \lhd \mathsf{U} \rhd y = \mathsf{U}, \tag{CP-U}$$

which implies $\mathsf{U} \mathbin{\between} x = \mathsf{U} \mathbin{\curlyvee} x = \mathsf{U}$ and $\neg\mathsf{U} = \mathsf{U}$, and also $\mathsf{F} \mathbin{\between} \mathsf{U} = \mathsf{F}$. Extending evaluation trees and the previous results on EqMSCL and MSCL to this setting turns out be rather simple. Finally, we define two short-circuit logics with undefinedness based on the extensions of CP and CP_{mem} with the axiom (CP-U).

Definition 7.1: The set $\mathcal{T}_A^{\mathsf{U}}$ of U-*evaluation trees* over A with leaves in $\{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ is defined inductively by

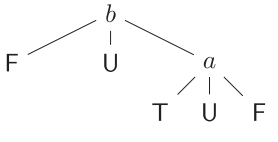
$$\mathsf{T} \in \mathcal{T}_A^{\mathsf{U}}, \quad \mathsf{F} \in \mathcal{T}_A^{\mathsf{U}}, \quad \mathsf{U} \in \mathcal{T}_A^{\mathsf{U}}, \quad (X \trianglelefteq \underline{a} \trianglerighteq Y) \in \mathcal{T}_A^{\mathsf{U}} \quad \text{for any } X, Y \in \mathcal{T}_A^{\mathsf{U}} \text{ and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

The operator $\_ \trianglelefteq \underline{a} \trianglerighteq \_$ is called U-*tree composition over a*. In the evaluation tree $X \trianglelefteq \underline{a} \trianglerighteq Y$, the root is represented by a, the left branch by X, the right branch by Y, and the underlining of the root represents a middle branch to the leaf U.

Next to the formal notation for evaluation trees, we again introduce a more pictorial representation. For example, the tree

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

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



(Picture 2)

An alternative representation of U-evaluation trees is obtained by replacing the middle branches of internal nodes by underlined versions of these nodes.

We extend the set $\mathcal{S}_A$ to $\mathcal{S}_A^{\mathsf{U}}$ of closed (sequential) propositional statements over A with U by the following grammar ($a \in A$):

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

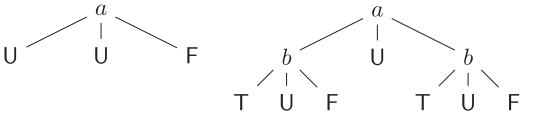
and refer to its signature by $\Sigma_{\mathsf{SCL}^{\mathsf{U}}}(A) = \{\wedge\!\!\!\circ, \vee\!\!\!\circ, \neg, \mathsf{T}, \mathsf{F}, \mathsf{U}, a \mid a \in A\}$.

We interpret propositional statements in $\mathcal{S}_A^{\mathsf{U}}$ as evaluation trees by extending the function *se* (Definition 2.2).

Definition 7.2: The unary *short-circuit evaluation function* $se^{\mathsf{U}} : \mathcal{S}_A^{\mathsf{U}} \to \mathcal{T}_A^{\mathsf{U}}$ is defined as follows, where $a \in A$:

$$\begin{aligned}
se^{\mathsf{U}}(\mathsf{T}) &= \mathsf{T}, \quad se^{\mathsf{U}}(\neg P) = se^{\mathsf{U}}(P)[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}],\\
se^{\mathsf{U}}(\mathsf{F}) &= \mathsf{F}, \quad se^{\mathsf{U}}(P \mathbin{\wedge\!\!\!\circ} Q) = se^{\mathsf{U}}(P)[\mathsf{T} \mapsto se^{\mathsf{U}}(Q)],\\
se^{\mathsf{U}}(\mathsf{U}) &= \mathsf{U}, \quad se^{\mathsf{U}}(P \mathbin{\vee\!\!\!\circ} Q) = se^{\mathsf{U}}(P)[\mathsf{F} \mapsto se^{\mathsf{U}}(Q)],\\
se^{\mathsf{U}}(a) &= \mathsf{T} \trianglelefteq \underline{a} \trianglerighteq \mathsf{F}.
\end{aligned}$$

Examples: $se^{\mathsf{U}}(a \mathbin{\wedge\!\!\!\circ} \mathsf{U}) = \mathsf{U} \trianglelefteq \underline{a} \trianglerighteq \mathsf{F}$ and $se^{\mathsf{U}}((a \mathbin{\vee\!\!\!\circ} \mathsf{T}) \mathbin{\wedge\!\!\!\circ} b) = (\mathsf{T} \trianglelefteq \underline{b} \trianglerighteq \mathsf{F}) \trianglelefteq \underline{a} \trianglerighteq (\mathsf{T} \trianglelefteq \underline{b} \trianglerighteq \mathsf{F})$ can be depicted as follows:



The extension to memorising evaluation trees in $\mathcal{T}_A^{\mathsf{U}}$ is straightforward (cf. Definition 3.1). For readability, we also use the name 'memorising evaluation trees' (rather than 'memorising U-evaluation trees').

Definition 7.3: The unary *memorising evaluation function*

$$mse^{\mathsf{U}} : \mathcal{S}_A^{\mathsf{U}} \to \mathcal{T}_A^{\mathsf{U}}$$

yields *memorising evaluation trees* and is defined by

$$mse^{\mathsf{U}}(P) = m^{\mathsf{U}}(se^{\mathsf{U}}(P)).$$

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

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

For $a \in A$, the auxiliary functions $L_a^{\mathsf{U}} : \mathcal{T}_A^{\mathsf{U}} \to \mathcal{T}_A^{\mathsf{U}}$ ('Left a-reduction') and $R_a^{\mathsf{U}} : \mathcal{T}_A^{\mathsf{U}} \to \mathcal{T}_A^{\mathsf{U}}$ ('Right a-reduction') are defined by

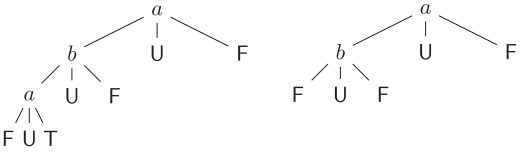
$$\begin{aligned}
L_a^{\mathsf{U}}(\mathsf{T}) = \mathsf{T}, \quad L_a^{\mathsf{U}}(\mathsf{F}) = \mathsf{F}, \quad L_a^{\mathsf{U}}(\mathsf{U}) = \mathsf{U},\\
L_a^{\mathsf{U}}(X \trianglelefteq \underline{b} \trianglerighteq Y) = \begin{cases} L_a^{\mathsf{U}}(X) & \text{if } b = a,\\ L_a^{\mathsf{U}}(X)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

and

$$R_a^{\mathsf{U}}(\mathsf{T}) = \mathsf{T}, \quad R_a^{\mathsf{U}}(\mathsf{F}) = \mathsf{F}, \quad R_a^{\mathsf{U}}(\mathsf{U}) = \mathsf{U},$$

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

As an example, we depict $se^{\mathsf{U}}(a \between (b \between \neg a))$ and the memorising evaluation tree $mse^{\mathsf{U}}(a \between (b \between \neg a))$:



We extend memorising *se*-congruence ($=_{mse}$) to $\mathcal{S}_A^{\mathsf{U}}$.

Definition 7.4: *Memorising se^{U}-congruence*, notation $=_{mse^{\mathsf{U}}}$, is defined on $\mathcal{S}_A^{\mathsf{U}}$ by

$$P =_{mse^{\mathsf{U}}} Q \iff mse^{\mathsf{U}}(P) = mse^{\mathsf{U}}(Q).$$

Both the extension of Lemma 3.5 to $\mathcal{T}_A^{\mathsf{U}}$ and the extension of the supporting Lemma 3.3 are trivial: all inductive proofs require only one extra, trivial base case.

Lemma 7.5: *The binary relation $=_{mse^{\mathsf{U}}}$ is a congruence on $\mathcal{S}_A^{\mathsf{U}}$.*

In Table 4, we extend the set of axioms EqMSCL to EqMSCL$^{\mathsf{U}}$ by adding the axiom (4). Defining $\mathsf{U}^{dl} = \mathsf{U}$ implies that EqMSCL$^{\mathsf{U}}$ satisfies the duality principle.

Theorem 7.6: *For all $P, Q \in \mathcal{S}_A^{\mathsf{U}}$, $\mathrm{EqMSCL}^{\mathsf{U}} \vdash P = Q \Rightarrow P =_{mse^{\mathsf{U}}} Q$.*

Proof: By Lemma 7.5 $=_{mse^{\mathsf{U}}}$ is a congruence on $\mathcal{S}_A^{\mathsf{U}}$. Clearly, $\neg\mathsf{U} =_{mse^{\mathsf{U}}} \mathsf{U}$. The remainder of the proof is equal to the proof of Theorem 4.1. ■

Some derivabilities in EqMSCL$^{\mathsf{U}}$:

$$\begin{aligned} \mathsf{U} \between \mathsf{F} &= (\mathsf{U} \between \mathsf{F}) \vee \mathsf{F} \quad \text{by (F5)} \\ &= (\mathsf{U} \vee (\mathsf{F} \vee \mathsf{F})) \between (\mathsf{U} \vee \mathsf{F}) \quad \text{by (Mem)}'\text{and (4)} \end{aligned}$$

Table 4. EqMSCL$^{\mathsf{U}}$, a set of axioms for memorising se^{U}-congruence over $\mathcal{S}_A^{\mathsf{U}}$.

$\mathsf{F} = \neg\mathsf{T}$	(Neg)
$x \vee y = \neg(\neg x \between \neg y)$	(Or)
$\mathsf{T} \between x = x$	(Tand)
$x \between (x \vee y) = x$	(Abs)
$(x \vee y) \between z = (\neg x \between (y \between z)) \vee (x \between z)$	(Mem)
$\neg\mathsf{U} = \mathsf{U}$	(Und)

$$= \mathsf{U}, \quad \text{by (F5) and Idempotence} \tag{7}$$

$$\begin{aligned}
\mathsf{U} \mathbin{\owedge} x &= (\mathsf{U} \mathbin{\ovee} \mathsf{F}) \mathbin{\owedge} x \quad \text{by (F5)} \\
&= (\mathsf{U} \mathbin{\owedge} (\mathsf{F} \mathbin{\owedge} x)) \mathbin{\ovee} (\mathsf{U} \mathbin{\owedge} x) \quad \text{by (Mem) and (Und)} \\
&= \mathsf{U} \mathbin{\ovee} (\mathsf{U} \mathbin{\owedge} x) \quad \text{by (F6) and (7)} \\
&= \mathsf{U}. \quad \text{by (Abs)}'
\end{aligned} \tag{8}$$

In order to prove the completeness of EqMSCL$^{\mathsf{U}}$, we again use normal forms.

Definition 7.7: *Memorising* SCL$^{\mathsf{U}}$ *Normal Forms* (mSUNFs) are inductively defined:

- $\mathsf{T}, \mathsf{F}, \mathsf{U}$ are mSUNFs , and
- $(a \mathbin{\owedge} P) \mathbin{\ovee} (\neg a \mathbin{\owedge} Q)$ is an mSUNF if $a \in A$, and P and Q are mSUNFs that do not contain a.

We write MSUNF for the set of all mSUNFs .

The following functions on MSUNF are used to compose mSUNFs .

Definition 7.8: For $a \in A$ the function $\mathsf{T}_a^{\mathsf{U}} : \text{MSUNF} \to \text{MSUNF}$ is defined by

$$\mathsf{T}_a^{\mathsf{U}}(\mathsf{T}) = \mathsf{T}, \quad \mathsf{T}_a^{\mathsf{U}}(\mathsf{F}) = \mathsf{F}, \quad \mathsf{T}_a^{\mathsf{U}}(\mathsf{U}) = \mathsf{U},$$

$$\mathsf{T}_a^{\mathsf{U}}((b \mathbin{\owedge} P_1) \mathbin{\ovee} (\neg b \mathbin{\owedge} P_2)) = \begin{cases} P_1 & \text{if } b = a, \\ (b \mathbin{\owedge} \mathsf{T}_a^{\mathsf{U}}(P_1)) \mathbin{\ovee} (\neg b \mathbin{\owedge} \mathsf{T}_a^{\mathsf{U}}(P_2)) & \text{otherwise.} \end{cases}$$

For $a \in A$ the function $\mathsf{F}_a^{\mathsf{U}} : \text{MSUNF} \to \text{MSUNF}$ is defined by

$$\mathsf{F}_a^{\mathsf{U}}(\mathsf{T}) = \mathsf{T}, \quad \mathsf{F}_a^{\mathsf{U}}(\mathsf{F}) = \mathsf{F}, \quad \mathsf{F}_a^{\mathsf{U}}(\mathsf{U}) = \mathsf{U},$$

$$\mathsf{F}_a^{\mathsf{U}}((b \mathbin{\owedge} P_1) \mathbin{\ovee} (\neg b \mathbin{\owedge} P_2)) = \begin{cases} P_2 & \text{if } b = a, \\ (b \mathbin{\owedge} \mathsf{F}_a^{\mathsf{U}}(P_1)) \mathbin{\ovee} (\neg b \mathbin{\owedge} \mathsf{F}_a^{\mathsf{U}}(P_2)) & \text{otherwise.} \end{cases}$$

So, $\mathsf{T}_a^{\mathsf{U}}$ removes the a-occurrences in an mSUNF under the assumption that a evaluates to T (and $\mathsf{F}_a^{\mathsf{U}}$ does this under the assumption that a evaluates to F). Note that for each $P \in \text{MSUNF}$, both $\mathsf{T}_a^{\mathsf{U}}(P)$ and $\mathsf{F}_a^{\mathsf{U}}(P)$ are also mSUNFs . In order to compose mSUNFs , we use the following lemma.

Lemma 7.9: *For all $a \in A$, $P \in \mathcal{S}_A^{\mathsf{U}}$, and $Q \in$* MSUNF ,

(1) EqMSCL$^{\mathsf{U}} \vdash a \mathbin{\owedge} (P \mathbin{\owedge} Q) = a \mathbin{\owedge} (P \mathbin{\owedge} \mathsf{T}_a^{\mathsf{U}}(Q))$,
(2) EqMSCL$^{\mathsf{U}} \vdash \neg a \mathbin{\owedge} (P \mathbin{\owedge} Q) = \neg a \mathbin{\owedge} (P \mathbin{\owedge} \mathsf{F}_a^{\mathsf{U}}(Q))$,
(3) EqMSCL$^{\mathsf{U}} \vdash a \mathbin{\owedge} (P \mathbin{\ovee} Q) = a \mathbin{\owedge} (P \mathbin{\ovee} \mathsf{T}_a^{\mathsf{U}}(Q))$,
(4) EqMSCL$^{\mathsf{U}} \vdash \neg a \mathbin{\owedge} (P \mathbin{\ovee} Q) = \neg a \mathbin{\owedge} (P \mathbin{\ovee} \mathsf{F}_a^{\mathsf{U}}(Q))$.

Proof: As the proof of Lemma 4.7 (in Appendix A.5), except that the inductive proofs require one more trivial base case. ■

Lemma 7.10: *For each* $P \in \mathcal{S}_A^{\mathsf{U}}$ *there is* $P' \in$ *MSUNF such that* $\mathrm{EqMSCL}^{\mathsf{U}} \vdash P = P'$.

Proof: As the proof of Lemma 4.8, except that the inductive proofs require one more trivial base case. ■

Theorem 7.11: *For all* $P, Q \in \mathcal{S}_A^{\mathsf{U}}$, $\mathrm{EqMSCL}^{\mathsf{U}} \vdash P = Q \iff P =_{mse^{\mathsf{U}}} Q$.

Proof: As the proof of Theorem 4.9, except that the inductive proof requires one more trivial base case. ■

Theorem 7.12: *The axioms of* $\mathrm{EqMSCL}^{\mathsf{U}}$ *are independent.*

Proof: See Appendix A.7. ■

We extend CP with axiom (CP-U) and write CP^{U} for this extension. Furthermore, we extend the function se^{U} (Definition 7.2) to closed terms over $\Sigma_{\mathrm{CP}^{\mathsf{U}}}(A)$ by

$$se^{\mathsf{U}}(P \trianglelefteq Q \trianglerighteq R) = se^{\mathsf{U}}(Q)[\,\mathsf{T} \mapsto se^{\mathsf{U}}(P),\ \mathsf{F} \mapsto se^{\mathsf{U}}(R)].$$

Proposition 7.13: *For all closed terms P, Q over* $\Sigma_{\mathrm{CP}^{\mathsf{U}}}(A)$,

$$\mathrm{CP}^{\mathsf{U}} \vdash P = Q \iff se^{\mathsf{U}}(P) = se^{\mathsf{U}}(Q).$$

Proof: See Appendix A.8. ■

Below we define short-circuit logics with 'undefinedness'.

Definition 7.14: A *short-circuit logic with undefinedness* is a logic that implies the consequences of the module expression

$$\mathrm{SCL}^{\mathsf{U}} = \{\,\mathsf{T}, \mathsf{U}, \neg, \mathbin{\diamondsuit\!\!\wedge}\,\} \,\square\, (\mathrm{CP}^{\mathsf{U}} \cup \{(\mathit{defNeg}), (\mathit{defAnd})\}).$$

We write $\mathrm{CP}^{\mathsf{U}}_{mem}$ for the extension of CP_{mem} with axiom (CP-U).

Definition 7.15: *Free short-circuit logic with undefinedness* ($\mathrm{FSCL}^{\mathsf{U}}$) is the short-circuit logic that implies no other consequences than those of the module expression $\mathrm{SCL}^{\mathsf{U}}$.

Memorising short-circuit logic with undefinedness ($\mathrm{MSCL}^{\mathsf{U}}$) is the short-circuit logic that implies no other consequences than those of the module expression

$$\{\,\mathsf{T}, \mathsf{U}, \neg, \mathbin{\diamondsuit\!\!\wedge}\,\} \,\square\, (\mathrm{CP}^{\mathsf{U}}_{mem} \cup \{(\mathit{defNeg}), (\mathit{defAnd})\}).$$

Of course, we intend to provide equational axiomatisations for both these short-circuit logics. For $\mathrm{FSCL}^{\mathsf{U}}$, we discuss a conjecture in Section 8, and for $\mathrm{MSCL}^{\mathsf{U}}$, we have a similar result as before (cf. Corollary 6.6).

Theorem 7.16: *For all terms s, t over* $\Sigma_{\mathrm{SCL}^{\mathsf{U}}}(A)$, $\mathrm{EqMSCL}^{\mathsf{U}} \vdash s = t \iff \mathit{MSCL}^{\mathsf{U}} \vdash s = t$.

Proof: The proof of Theorem 6.5 as well as the proofs of all supporting lemmas can be easily extended to the signatures containing U. ■

Theorems 7.11 and 7.16 imply the following result.

Corollary 7.17: *MSCL*$^{\mathsf{U}}$ *axiomatises memorising se*^{U}*-congruence* $=_{mse^{\mathsf{U}}}$*, and* EqMSCL$^{\mathsf{U}}$ *is an equational axiomatisation of MSCL*$^{\mathsf{U}}$.

8. Discussion

In this section, we discuss some other variants of sequential connectives. Next, we relate MSCL$^{\mathsf{U}}$ to McCarthy's three-valued logic and observe that we cannot relate FSCL$^{\mathsf{U}}$ to any well-known three-valued logic. We conclude with a note on two-valued SSCL (which comprises an axiomatisation).

Boolean connectives that prescribe short-circuit evaluation often have specific names or notations, for example, Dijkstra's *cand* (conditional conjunction) and *cor* used in the three-valued setting with *undefined* (see Dijkstra, 1976; Gries, 1981), or the short-circuited connectives && and || as used in programming languages such as C, Go, Java, and Perl. Short-circuit evaluation in C is discussed in e.g. Zimmermann and Dold (2003). Other notations for the sequential connectives $\mathbin{\owedge}$ and $\mathbin{\ovee}$ with memorising interpretation are $\triangle$ and $\triangledown$ from computability logic (see, e.g. Japaridze, 2008), and $\otimes$ and $\oplus$ from transaction logic (see, e.g. Basseda & Kifer, 2015, there called *serial* connectives). However, MSCL is just a part of both these logics and it is questionable whether its axiomatisation or semantics are of any relevance.

In Guzmán and Squier (1990), so-called *Conditional logic* is defined and axiomatised. This logic is named after the discussion in Gries (1981, pp. 68–70) about logical laws for the connectives *cand* and *cor*. Some typical equations that hold in conditional logic, where *cand* and *cor* are written as $\wedge$ and $\vee$, are

$$\neg \mathsf{U} = \mathsf{U}, \quad \mathsf{U} \wedge x = \mathsf{U}, \quad \mathsf{F} \wedge x = \mathsf{F}, \quad \text{and} \tag{9}$$

$$(x \wedge y) \vee (y \wedge x) = (y \wedge x) \vee (x \wedge y). \tag{10}$$

However, it is not so clear whether (Guzmán & Squier, 1990) follow the intentions of Gries (1981). In the words of Pigozzi (1990):[5]

> Consider now the protomatrix MC_3 of McCarthy's noncommuting conditional logic. Gries seems clearly to have an assertional logic in mind when he deals with this protomatrix in Gries (1981). On the other hand, in their detailed study of the logic MC_3 carried out in 1990, Guzmán and Squier deal exclusively with the equational logic. The exact connection between the assertional and the equational case is not clear, but it is certainly not as strong as in the classical (Boolean) case.

When the binary connectives in (9) and (10) are replaced by $\mathbin{\owedge}$ and $\mathbin{\ovee}$, equations (9) are consequences in MSCL$^{\mathsf{U}}$, but (10) is not: a memorising evaluation tree with root *a* is not equal to one with root *b*. In fact, omitting (10) from the axiomatisation provided in Guzmán and Squier (1990) yields an axiomatisation as strong as EqMSCL$^{\mathsf{U}}$, which follows easily with *Prover9* (McCune, 2008). Thus, MSCL$^{\mathsf{U}}$ is a three-valued logic different from conditional logic and is new, as far as we know. We refer to Bergstra et al. (1995)

Table 5. An alternative set of CP-axioms for defining SSCL.

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

for an overview of three-valued logics and their axiomatisations, including conditional logic (there called McCarthy's three-valued logic, as in Nagata et al., 1975).

We did not find any work related to the three-valued logic $\mathrm{FSCL}^{\mathsf{U}}$ (Definition 7.15). It seems that extending the equational axiomatisation EqFSCL with the axiom $\mathsf{U} \owedge x = \neg\mathsf{U}$, say $\mathrm{EqFSCL}^{\mathsf{U}}$, is an attractive option: $\mathrm{EqFSCL}^{\mathsf{U}}$ implies $\neg\mathsf{U} = \mathsf{U}$ (and thus $\mathsf{U} \owedge x = \mathsf{U}$ and $\mathsf{U} \owedge x = \mathsf{U}$) and we could not find any desirable equation that is non-derivable. Furthermore, if (F1) and (F3) are omitted, $\mathrm{EqFSCL}^{\mathsf{U}}$ is independent (compare Fact 2.5). We formulate the following conjecture (see also challenge (2) in Section 9).

Conjecture 8.1: For all $P, Q \in \mathcal{S}_A^{\mathsf{U}}$, $\mathrm{FSCL}^{\mathsf{U}} \vdash P = Q \iff \mathrm{EqFSCL}^{\mathsf{U}} \vdash P = Q$.

We conclude this section with a few words on the definition of two-valued static short-circuit logic SSCL (Definition 5.2). In Table 5, we provide an alternative set of axioms for defining SSCL that is not a simple extension of CP or CP_{mem}. Note that axiom (CP3s) with $y = \mathsf{T}$ implies (CP3), and with $y = \mathsf{F}$ implies $\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}$ (the axiom that is used to define SSCL). These axioms are independent (which easily follows with *Mace4*, McCune, 2008). A proof of one of the axioms (CPmem1) or (CPmem3) by *Prover9* (McCune, 2008) 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 axiomatisation of static *se*-congruence is not a goal of this paper: the axiomatisation in Definition 5.2 is sufficiently simple and expresses the fundamental intuitions appropriately. Reasons to present the axiomatisation in Table 5 are its independence (contrary to $\mathrm{CP}_{mem} \cup \{\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}\}$) and, of course, its striking simplicity (cf. Hoare, 1985).

9. Conclusions

In Bergstra and Ponse (2011), we introduced 'proposition algebra', which is based on Hoare's conditional $x \lhd y \rhd z$ and the constants T and 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 axiomatisations of the resulting 'valuation congruences': CP (four axioms) axiomatises free valuation congruence (the least identifying valuation congruence we consider), and the extension CP_{mem} (one extra axiom) axiomatises memorising valuation congruence, the most identifying valuation congruence below 'sequential propositional logic'. Static valuation congruence

can be axiomatised by adding the axiom $\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}$ to CP_{mem}, and can be seen as an axiomatisation of sequential propositional logic.

In Bergstra and Ponse (2010, 2012), we introduced an alternative valuation semantics for proposition algebra in the form of *Hoare-McCarthy algebras* (HMAs) that is more elegant than the semantic framework provided in Bergstra and Ponse (2011): HMA-based semantics has the advantage that one can define a valuation congruence without first defining the valuation *equivalence* it is contained in.

In Bergstra and Ponse (2017), following the approach of Staudt (2012), we defined evaluation trees as a simpler and more direct semantics for proposition algebra and proved several completeness results for the valuation congruences mentioned above.

In Bergstra et al. (2013), we introduced 'short-circuit logic' as defined here (Definitions 5.1 and 5.2). In Ponse and Staudt (2018), summarised in Section 2, we dealt with the case of free short-circuit logic (FSCL).

In this paper, we have shown that memorising short-circuit logic can be understood and used without any reference to (or dependence on) the conditional connective, in spite of the fact that it is defined with this connective: MSCL can also be seen as the equational logic defined by EqMSCL, and with equality of memorising evaluation trees as a simple semantics (Theorem 4.9). The meaning of a sequential proposition is determined by the entire *process* of its sequential evaluation, as modelled by its memorising evaluation tree. This perspective explains why the sequential connectives are taken to be non-commutative and why the constants T and F are not definable (and thus included). In Section 4, we discussed the following properties of MSCL:

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

$$x \owedge (y \ovee z) = (x \owedge y) \ovee (x \owedge z), \quad \text{left} - \text{distributivity(LD)}$$
$$x \owedge (y \owedge x) = x \owedge y. \quad \text{(see (C2) for a proof)}$$

  None of these hold in FSCL.

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

$$(x \owedge y) \ovee (\neg x \owedge z) = (\neg x \ovee y) \owedge (x \ovee z), \tag{M1}$$
$$(x \owedge y) \ovee (\neg x \owedge z) = (\neg x \owedge z) \ovee (x \owedge y), \tag{M2}$$

and the right-distributivity of $\owedge$ over '$\mathtt{if}\ x\ \mathtt{then}\ y\ \mathtt{else}\ z$', that is

$$(\mathtt{if}\, x\, \mathtt{then}\, y\, \mathtt{else}\, z) \owedge u = \mathtt{if}\, x\, \mathtt{then}\, (y \owedge u)\, \mathtt{else}\, (z \owedge u),$$

which is characterised by

$$((x \owedge y) \ovee (\neg x \owedge z)) \owedge u = (x \owedge (y \owedge u)) \ovee (\neg x \owedge (z \owedge u)). \tag{M3}$$

In a similar way, the right-distributivity of $\ovee$ over '$\mathtt{if}\ x\ \mathtt{then}\ y\ \mathtt{else}\ z$' follows.

In Sections 1 and 5, we briefly discussed static short-circuit logic: SSCL defines a version of ordinary propositional logic (PL) by means of sequential connectives that prescribe short-circuit evaluation *and* that are commutative. So, any equational axiomatisation of PL is one for SSCL, provided the connectives are replaced by their sequential counterparts (and $x \circ\!\!\rightarrow y = \neg x \mathbin{\diamond\!\!\vee} y$). However, we think SSCL does not yield interesting perspectives and is counterintuitive from a programming-oriented point of view. For example, the identity $a \mathbin{\diamond\!\!\wedge} b = b \mathbin{\diamond\!\!\wedge} a$ requires that their memorising evaluation trees should be considered equal. This either requires a transformation of memorising evaluation trees according to some fixed ordering of atoms (comprising a and b), which may not agree with the actual evaluation order, or an equivalence relation on memorising evaluation trees that does not respect the evaluation order of atoms.[6]

We conclude that MSCL provides a more natural view on sequential propositional logic than SSCL.

In Section 7, we extended our main results to a three-valued setting by including a constant U that represents McCarthy's notion of undefinedness (Theorems 7.11 and 7.16). The extension to evaluation trees with leaves in $\{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ is straightforward, and so is their memorising variant. The resulting short-circuit logic $\mathrm{MSCL}^{\mathsf{U}}$ satisfies the duality principle, has a simple equational axiomatisation, and preserves all MSCL-identities. We also introduced $\mathrm{FSCL}^{\mathsf{U}}$, i.e. three-valued free short-circuit logic, and raised the question of a complete equational axiomatisation for closed terms (Conjecture 8.1). Finally, as observed in Section 1, two-valued SSCL cannot be extended in this way because commutativity of the sequential connectives would imply that $\mathsf{U} = \mathsf{F}$.

In the Introduction, we distinguished short-circuit logics (SCLs) into two- or three-valuedness. Another, more programmer-oriented criterion for distinction is whether or not atomic side effects can occur in the evaluation of a conditional expression. If so, then FSCL (or $\mathrm{FSCL}^{\mathsf{U}}$) is the appropriate SCL; a typical example (allowed in many programming languages) is a condition containing an assignment (as an atom) and some examples are discussed in Ponse and Staudt (2018, Section 4.4). For the case where atoms (and hence conditions) are 'side effect free', MSCL or $\mathrm{MSCL}^{\mathsf{U}}$ is the appropriate logic to reason about their logical equivalences, and the choice of which of the two is determined only by whether the atoms involved can yield the truth value *undefined*. A sequential version of the second example in the Introduction, viz,

$$(x \neq 0) \mathbin{\diamond\!\!\wedge} (y/x > 17),$$

is of course a typical example for using $\mathrm{MSCL}^{\mathsf{U}}$. Other examples are mentioned in Gries (1981) and concern for example arrays that are indexed out of their bounds.

Challenging questions and future work. A challenging question with respect to the proof of Theorem 4.2, that is, $\mathrm{EqMSCL} \vdash \mathrm{EqFSCL}$, is to find a shorter and more comprehensible proof of associativity. Alternatively, find another equational axiomatisation of MSCL that is short and simple, uses only three variables, and admits a simple proof of this theorem.

Two other challenges, following the discussion in Section 8:

(1) Find a convincing example that distinguishes $\mathrm{MSCL}^{\mathsf{U}}$ from Conditional Logic as defined in Guzmán and Squier (1990).

(2) Solve Conjecture 8.1, i.e. find out whether EqFSCL$^{\mathsf{U}}$ axiomatises FSCL$^{\mathsf{U}}$ for closed terms. And, if this is the case, find out whether EqFSCL$^{\mathsf{U}}$ and FSCL$^{\mathsf{U}}$ define the same equational theory.[7] We note that we could not transfer the completeness proofs in this paper to FSCL$^{\mathsf{U}}$: we could not find suitable normal forms (cf. Ponse & Staudt, 2018, Sect.2.2), and the conditional $x \lhd y \rhd z$ is not definable in FSCL$^{\mathsf{U}}$ (cf. Bergstra & Ponse, 2011, Prop.12.1).

In future work, we want to cover two more short-circuit logics, 'repetition-proof' and 'contractive' short-circuit logic (see Bergstra et al., 2013), and to provide examples of program fragments that illustrate all short-circuits logics defined (compare Ponse & Staudt, 2018, Sect.4.4).

Notes

1. In Staudt (2012), the dual of axiom (F5) is used, and in Ponse and Staudt (2018), *se*-congruence is called 'free valuation congruence'.
2. See https://oeis.org/A065410.
3. For the conditioned disjunction, reference (Church, 1956) is often used, and also the name *conditional disjunction*.
4. In 1963, Dicker provided in (Dicker, 1963) a set of five independent and simple axioms for the conditioned disjunction. In Section 8 we provide a set of four independent and simple axioms that is also complete.
5. The *protomatrix* mentioned is the set of truth tables for $\{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$.
6. In Bergstra and Ponse (2017), we defined static evaluation trees for $\mathrm{CP}_{mem} \cup \{\mathsf{F} \lhd x \rhd \mathsf{F} = \mathsf{F}\}$ according to the first requirement.
7. This last challenge generalises an open question about EqFSCL from Ponse and Staudt (2018): *For open terms s, t over* $\Sigma_{\mathrm{SCL}}(A)$, *can* $\mathrm{EqFSCL} \vdash s = t \iff \mathrm{FSCL} \vdash s = t$ *be proved?*
8. We speak of 'basic forms' instead of normal forms in order to avoid intuitions from term rewriting: for example, the basic form associated with action a is $\mathsf{T} \lhd a \rhd \mathsf{F}$, whereas one would expect that the normal form of the latter is a.
9. Without loss of generality it can be assumed that substitutions happen first in equational proofs (see, e.g. Aceto et al., 2008).

Acknowledgments

We thank an anonymous reviewer for his helpful comments and, in particular, for his suggestion to consider three-valued cases of short-circuit logic as well.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Jan A. Bergstra http://orcid.org/0000-0003-2492-506X
Alban Ponse http://orcid.org/0000-0001-6061-5355

References

Aceto L., Chen T., Fokkink W. J., & Ingolfsdottir A. (2008). On the axiomatizability of priority. Mathematical Structures in Computer Science, *18*(1), 5–28. https://doi.org/10.1017/S0960129507006524

Basseda R., & Kifer M. (2015). Planning with regression analysis in transaction logic. In B. ten Cate, & A. Mileo (Eds.), *RR 2015: Web reasoning and rule systems* (pp. 45–60). LNCS 9209. Springer. https://doi.org/10.1007/978-3-319-22002-4_5

Bergstra J. A., Bethke I., & 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. https://doi.org/10.1080/11663081.1995.10510855

Bergstra J. A., Heering J., & Klint P. (1990). Module algebra. Journal of the ACM, *37*(2), 335–372. https://doi.org/10.1145/77600.77621

Bergstra J. A., & Ponse A. (2010). *On Hoare-McCarthy algebras*. http://arxiv.org/abs/1012.5059 [cs.LO].

Bergstra J. A., & Ponse A. (2011). Proposition algebra. ACM Transactions on Computational Logic, *12*(3), Article 21, 1–36. https://doi.org/10.1145/1929954.1929958

Bergstra J. A., & Ponse A. (2012). Proposition algebra and short-circuit logic. In F. Arbab & M. Sirjani (Eds.), *Proceedings of the 4th international conference on fundamentals of software engineering* (FSEN 2011, Tehran) (pp. 15–31). LNCS 7141. Springer. https://doi.org/10.1007/978-3-642-29320-7_2

Bergstra J. A., & Ponse A. (2017). *Evaluation trees for proposition algebra*. https://arxiv.org/abs/1504.08321v3 [cs.LO].

Bergstra J. A., Ponse A., & Staudt D. J. C. (2013). *Short-circuit logic*. https://arxiv.org/abs/1010.3674v4 [cs.LO, math.LO].

Church A. (1948). Conditioned disjunction as a primitive connective for the propositional calculus. Portugaliae Mathematica, *7* (2), 87–90.

Church A (1956). Introduction to mathematical logic. Princeton University Press.

Dicker R. M. (1963). A set of independent axioms for Boolean algebra. Proceedings of the London Mathematical Society, *s3-13*(1), 20–30. https://doi.org/10.1112/plms/s3-13.1.20

Dijkstra E. W. (1976). A discipline of programming. Prentice Hall, Inc.

Gries D. (1981). The science of programming. Springer-Verlag.

Guzmán F., & Squier C. C. (1990). The algebra of conditional logic. Algebra Universalis, *27*(1), 88–110. https://doi.org/10.1007/BF01190256

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. https://doi.org/10.1002/(ISSN)1521-3870

Japaridze G. (2008). Sequential operators in computability logic. Information and Computation, *206*(12), 1443–1475. https://doi.org/10.1016/j.ic.2008.10.001

McCarthy J. (1963). A basis for a mathematical theory of computation. In P. Braffort, & D. Hirschberg (Eds.), *Computer programming and formal systems* (pp. 33–70). Volume 35 of Studies in logic and the foundations of mathematics. Elsevier.

McCune W. (2008). The GUI: Prover9 and Mace4 with a graphical user interface. Prover9-Mace4-v05B.zip. Retrieved March 14, 2008, from https://www.cs.unm.edu/mccune/prover9/gui/v05.html

Moret B. M. E. (1982). Decision trees and diagrams. Computing Surveys, *14*(4), 593–623. https://doi.org/10.1145/356893.356898

Nagata M., Nakanishi M., & Nishimura T. (1975). Implementation of Lukasiewicz's, Kleene's and McCarthy's 3-valued logic. Science Reports of the Tokyo Kyoiku Daigaku, Section A, *13*(347–365), 90–100.

Pigozzi Don L. (1990). Data types over multiple-valued logics. Theoretical Computer Science, *77*(1–2), 161–194. https://doi.org/10.1016/0304-3975(90)90119-3

Ponse A., & Staudt D. J. C. (2018). An independent axiomatisation for free short-circuit logic. Journal of Applied Non-Classical Logics, *28*(1), 35–71. https://doi.org/10.1080/11663081.2018.1448637

Staudt D. J. C. (2012, May). *Completeness for two left-sequential logics* [MSc. thesis Logic, University of Amsterdam]. https://arxiv.org/abs/1206.1936 [cs.LO].

Zimmermann W., & Dold A. (2003). A framework for modeling the semantics of expression evaluation with abstract state machines. In E. Börger, A. Gargantini, & E. Riccobene (Eds.), *ASM 2003* (pp. 391–406). LNCS 2589. Springer. https://doi.org/10.1007/3-540-36498-6_23

Appendix. Detailed proofs

A.1 A proof of Lemma 3.3

Lemma 3.3: *For all* $a, b \in A$, $f, f' \in \{L, R\}$, *and* $X, Y \in \mathcal{T}_A$,

(1) *If* $b \neq a$ *then* $f_a(f'_b(X)) = f'_b(f_a(X))$,
(2) $f_a(m(f_a(X))) = f_a(m(X))$,
(3) $f_a(m(X)) = m(f_a(X))$,
(4) $m(f_a(X[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}])) = m(f_a(X))[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}]$,
(5) $f_a(X[\mathsf{T} \mapsto Y]) = f_a(X)[\mathsf{T} \mapsto f_a(Y)]$,
(6) *If* $m(X) = m(Y)$ *then* $m(f_a(X)) = m(f_a(Y))$,
(7) $m(f_a(m(X))) = m(f_a(X))$.

Proof: (1) By structural induction on X. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ then distinguish the cases $c = a$ and $c \notin \{a, b\}$:

Case $c = a$, subcase $f = f' = L$. Then $L_a(L_b(X)) = L_a(L_b(X_1) \trianglelefteq a \trianglerighteq L_b(X_2)) = L_a(L_b(X_1))$ and $L_b(L_a(X)) = L_b(L_a(X_1))$, so by induction we are done.

Subcase $f = L$ and $f' = R$. Then $L_a(R_b(X)) = L_a(R_b(X_1) \trianglelefteq a \trianglerighteq R_b(X_2)) = L_a(R_b(X_1))$ and $R_b(L_a(X)) = R_b(L_a(X_1))$, so by induction we are done.

The remaining subcases follow in a similar way.

Case $c \notin \{a, b\}$. All subcases for f and f' follow easily by induction.

(2) ($f_a(m(f_a(X))) = f_a(m(X))$.) By induction on $d(X)$. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ then distinguish the cases $c = a$ and $c \neq a$:

Case $c = a$, subcase $f = L$. Then

$$\begin{aligned} L_a(m(L_a(X))) &= L_a(m(L_a(X_1 \trianglelefteq a \trianglerighteq X_2))) \\ &= L_a(m(L_a(X_1))) \\ &= L_a(m(L_a(X_1)) \trianglelefteq a \trianglerighteq m(R_a(X_2))) \\ &= L_a(m(X)), \end{aligned}$$

so induction is not needed in this case. The remaining subcase follows in a similar way.

Case $c \neq a$, subcase $f = L$. Then

$$\begin{aligned} L_a(m(L_a(X))) &= L_a(m(L_a(X_1 \trianglelefteq c \trianglerighteq X_2))) \\ &= L_a(m(L_c(L_a(X_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2)))) \\ &= L_a(m(L_c(L_a(X_1)))) \trianglelefteq c \trianglerighteq L_a(m(R_c(L_a(X_2)))) \\ &= L_a(m(L_a(L_c(X_1)))) \trianglelefteq c \trianglerighteq L_a(m(L_a(R_c(X_2)))) \quad \text{by La.3.3.1} \\ &= L_a(m(L_c(X_1))) \trianglelefteq c \trianglerighteq L_a(m(R_c(X_2))) \quad \text{by IH} \\ &= L_a(m(L_c(X_1)) \trianglelefteq c \trianglerighteq m(R_c(X_2))) \\ &= L_a(m(X)). \end{aligned}$$

Note that $d(L_c(X_1)) \leq d(X_1) < d(X)$ and $d(R_c(X_2)) \leq d(X_2) < d(X)$. The remaining subcase follows in a similar way.

(3) ($f_a(m(X)) = m(f_a(X))$.) By induction on $d(X)$. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ distinguish the cases $c = a$ and $c \neq a$:

Case $c = a$, subcase $f = L$. Then

$$\begin{aligned} L_a(m(X)) &= L_a(m(L_a(X_1)) \trianglelefteq a \trianglerighteq m(R_a(X_2))) \\ &= L_a(m(L_a(X_1))) \\ &= L_a(m(X_1)) \quad \text{by La.3.3.2} \\ &= m(L_a(X_1)) \quad \text{by IH} \\ &= m(L_a(X_1 \trianglelefteq a \trianglerighteq X_2)) \\ &= m(L_a(X)). \end{aligned}$$

The remaining subcase follows in a similar way.

Case $c \neq a$, subcase $f = L$. Then

$$\begin{aligned} L_a(m(X)) &= L_a(m(L_c(X_1)) \trianglelefteq c \trianglerighteq m(R_c(X_2))) \\ &= L_a(m(L_c(X_1))) \trianglelefteq c \trianglerighteq L_a(m(R_c(X_2))) \\ &= m(L_a(L_c(X_1))) \trianglelefteq c \trianglerighteq m(L_a(R_c(X_2))) \quad \text{by IH} \\ &= m(L_c(L_a(X_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2))) \quad \text{by La.3.3.1} \\ &= m(L_a(X_1) \trianglelefteq c \trianglerighteq L_a(X_2)) \\ &= m(L_a(X)). \end{aligned}$$

Note that $d(L_c(X_1)) \leq d(X_1) < d(X)$ and $d(R_c(X_2)) \leq d(X_2) < d(X)$. The remaining subcase follows in a similar way.

(4) ($m(f_a(X[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}])) = m(f_a(X))[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}]$.) By induction on $d(X)$. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ distinguish the cases $c = a$ and $c \neq a$:

Case $c = a$, subcase $f = L$. Then $L_a(X) = L_a(X_1)$ and by induction we are done. The remaining subcase follows in a similar way.

Case $c \neq a$, subcase $f = L$. Write [neg] for the leaf replacement $[\mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}]$. Then

$$\begin{aligned} m(L_a(X[\text{neg}])) &= m(L_c(L_a(X_1[\text{neg}]))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2[\text{neg}]))) \\ &= m(L_a(L_c(X_1[\text{neg}]))) \trianglelefteq c \trianglerighteq m(L_a(R_c(X_2[\text{neg}]))) \quad \text{by La.3.3.1} \\ &= m(L_a(L_c(X_1)))[\text{neg}] \trianglelefteq c \trianglerighteq m(L_a(R_c(X_2)))[\text{neg}] \quad \text{by IH} \\ &= m(L_c(L_a(X_1)))[\text{neg}] \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2)))[\text{neg}] \quad \text{by La.3.3.1} \\ &= (m(L_c(L_a(X_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2))))[\text{neg}] \\ &= m(L_a(X))[\text{neg}]. \end{aligned}$$

The remaining subcase follows in a similar way.

(5) ($f_a(X[\mathsf{T} \mapsto Y]) = f_a(X)[\mathsf{T} \mapsto f_a(Y)]$.) By induction on the structure of X. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ distinguish the cases $c = a$ and $c \neq a$:

Case $c = a$, subcase $f = L$. Then $L_a(X[\mathsf{T} \mapsto Y]) = L_a(X_1[\mathsf{T} \mapsto Y])$ and by induction $L_a(X_1[\mathsf{T} \mapsto Y]) = L_a(X_1)[\mathsf{T} \mapsto Y] = L_a(X)[\mathsf{T} \mapsto Y]$. The remaining subcase follows in a similar way.

Case $c \neq a$, subcase $f = L$: $L_a(X[\mathsf{T} \mapsto Y]) = L_a(X_1[\mathsf{T} \mapsto Y]) \trianglelefteq c \trianglerighteq L_a(X_2[\mathsf{T} \mapsto Y]) \overset{IH}{=} L_a(X_1)[\mathsf{T} \mapsto L_a(Y)] \trianglelefteq c \trianglerighteq L_a(X_2)[\mathsf{T} \mapsto L_a(Y)] = L_a(X)[\mathsf{T} \mapsto L_a(Y)]$. The remaining subcase follows in a similar way.

(6) (If $m(X) = m(Y)$ then $m(f_a(X)) = m(f_a(Y))$.) By induction on $d(X)$. The base cases are trivial, and if $X = X_1 \trianglelefteq c \trianglerighteq X_2$ distinguish the cases $c = a$ and $c \neq a$:

Case $c = a$, subcase $f = L$. Then $m(L_a(X)) = m(L_a(X_1))$ and $m(R_a(X)) = m(R_a(X_2))$, and $m(X) = m(Y)$ implies that $Y = Y_1 \trianglelefteq a \trianglerighteq Y_2$, so $m(f_a(X)) = m(f_a(Y))$. The remaining subcase follows in a similar way.

Case $c \neq a$, subcase $f = L$. Then $m(X) = m(Y)$ implies that $Y = Y_1 \trianglelefteq c \trianglerighteq Y_2$, and $m(L_c(X_1)) = m(L_c(Y_1))$ and $m(R_c(X_2)) = m(R_c(Y_2))$. Derive

$$\begin{aligned} m(L_a(X)) &= m(L_c(L_a(X_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2))) \\ &= m(L_a(L_c(X_1))) \trianglelefteq c \trianglerighteq m(L_a(R_c(X_2))) \quad \text{by La.3.3.1} \\ &= m(L_a(L_c(Y_1))) \trianglelefteq c \trianglerighteq m(L_a(R_c(Y_2))) \quad \text{by IH} \\ &= m(L_c(L_a(Y_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(Y_2))) \quad \text{by La.3.3.1} \\ &= m(L_a(Y)). \end{aligned}$$

The remaining subcase follows in a similar way.

(7) $(m(f_a(m(X))) = m(f_a(X)).)$ By induction on $d(X)$. The base cases are trivial. If $X = X_1 \trianglelefteq c \trianglerighteq X_2$ then distinguish the cases $c = a$ and $c \neq a$:

Case $c = a, f = L$:

$$\begin{aligned} m(L_a(m(X))) &= m(L_a(m(X_1 \trianglelefteq a \trianglerighteq X_2))) \\ &= m(L_a(m(L_a(X_1)) \trianglelefteq a \trianglerighteq m(R_a(X_2)))) \\ &= m(L_a(m(L_a(X_1)))) \\ &= m(L_a(m(X_1))) \quad \text{by La.3.3.2} \\ &= m(L_a(X1)) \quad \text{by IH} \\ &= m(L_a(X)). \end{aligned}$$

The remaining subcase $f = R$ follows in a similar way.

Case $c \neq a$, subcase $f = L$:

$$\begin{aligned} m(L_a(m(X))) &= m(L_a(m(X_1 \trianglelefteq c \trianglerighteq X_2))) \\ &= m(L_a(m(L_c(X_1)) \trianglelefteq c \trianglerighteq m(R_c(X_2)))) \\ &= m(L_a(m(L_c(X_1))) \trianglelefteq c \trianglerighteq L_a(m(R_c(X_2)))) \\ &= m(L_c(L_a(m(L_c(X_1))))) \trianglelefteq c \trianglerighteq m(R_c(L_a(m(R_c(X_2)))))) \\ &= m(L_a(L_c(m(L_c(X_1))))) \trianglelefteq c \trianglerighteq m(L_a(R_c(m(R_c(X_2)))))) \quad \text{by La.3.3.1} \\ &= m(L_a(L_c(m(X_1)))) \trianglelefteq c \trianglerighteq m(L_a(R_c(m(X_2)))) \quad \text{by La.3.3.2} \\ &= m(L_a(m(L_c(X_1)))) \trianglelefteq c \trianglerighteq m(L_a(m(R_c(X_2)))) \quad \text{by La.3.3.3} \\ &= m(L_a(L_c(X_1))) \trianglelefteq c \trianglerighteq m(L_a(R_c(X_2))) \quad \text{by IH} \\ &= m(L_c(L_a(X_1))) \trianglelefteq c \trianglerighteq m(R_c(L_a(X_2))) \quad \text{by La.3.3.1} \\ &= m(L_a(X_1) \trianglelefteq c \trianglerighteq L_a(X_2)) \\ &= m(L_a(X)). \end{aligned}$$

Note that $d(L_c(X_1)) \leq d(X_1) < d(X)$ and $d(R_c(X_2)) \leq d(X_2) < d(X)$. The remaining subcase $f = R$ follows in a similar way. ■

A.2 Continuation of the proof of Theorem 4.1

Theorem 4.1: *For all* $P, Q \in \mathcal{S}_A$, EqMSCL $\vdash P = Q \Rightarrow P =_{mse} Q$.

Proof: The case for axiom (Mem). First derive

$$(X \mathbin{\widetilde{\triangledown}} Y) \mathbin{\widetilde{\wedge}} Z = X[\mathsf{F} \mapsto Y][\mathsf{T} \mapsto Z]$$

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

and

$$\begin{aligned}(\rightharpoonup X \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X \mathbin{\tilde{\wedge}} Z) &= (X[\mathsf{T} \mapsto \ \mathsf{F},\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]])[\mathsf{F} \mapsto X[\mathsf{T} \mapsto Z]] \\ &= X[\mathsf{T} \mapsto X[\mathsf{T} \mapsto Z],\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z[\mathsf{F} \mapsto X[\mathsf{T} \mapsto Z]]]]. \qquad \text{(R)}\end{aligned}$$

It suffices to show that for all $X, Y, Z \in \mathcal{T}_A$, $m((X \mathbin{\tilde{\vee}} Y) \mathbin{\tilde{\wedge}} Z) = m((\rightharpoonup X \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X \mathbin{\tilde{\wedge}} Z))$, which follows by structural induction on X. The base cases are trivial.

If $X = X_1 \trianglelefteq a \trianglerighteq X_2$, we find by (L) that

$$\begin{aligned}&m((X \mathbin{\tilde{\vee}} Y) \mathbin{\tilde{\wedge}} Z) \\ &\quad = m(L_a(X_1[\mathsf{T} \mapsto Z,\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]])) \trianglelefteq a \trianglerighteq m(R_a(X_2[\mathsf{T} \mapsto Z,\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]])).\end{aligned}$$

Furthermore, let $[Rep] = [\mathsf{T} \mapsto X[\mathsf{T} \mapsto Z],\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z[\mathsf{F} \mapsto X[\mathsf{T} \mapsto Z]]]]$, then

$$m((\rightharpoonup X \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X \mathbin{\tilde{\wedge}} Z)) = m(L_a(X_1[Rep])) \trianglelefteq a \trianglerighteq m(R_a(X_2[Rep])). \tag{Aux3}$$

With Lemma 3.3.5 it follows that (cf. (Aux2))

$$L_a(X_1[Rep]) = L_a(X_1[\mathsf{T} \mapsto X_1[\mathsf{T} \mapsto Z],\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z[\mathsf{F} \mapsto X_1[\mathsf{T} \mapsto Z]]]]), \tag{RepL}$$

$$R_a(X_2[Rep]) = R_a(X_2[\mathsf{T} \mapsto X_2[\mathsf{T} \mapsto Z],\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z[\mathsf{F} \mapsto X_2[\mathsf{T} \mapsto Z]]]]). \tag{RepR}$$

We derive

$$\begin{aligned}&m(L_a(X_1[\mathsf{T} \mapsto Z,\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]])) \\ &\quad = m(L_a(m(X_1[\mathsf{T} \mapsto Z,\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]]))) \quad \text{by La.3.3.7} \\ &\quad = m(L_a(m((X_1 \mathbin{\tilde{\vee}} Y) \mathbin{\tilde{\wedge}} Y))) \quad \text{by (L)} \\ &\quad = m(L_a(m((\rightharpoonup X_1 \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X_1 \mathbin{\tilde{\wedge}} Z)))) \quad \text{by IH} \\ &\quad = m(L_a((\rightharpoonup X_1 \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X_1 \mathbin{\tilde{\wedge}} Z))) \quad \text{by La.3.3.7} \\ &\quad = m(L_a(X_1[Rep])). \quad \text{by (R) and (RepL)}\end{aligned}$$

In a similar way it follows with (RepR) that

$$m(R_a(X_2[\mathsf{T} \mapsto Z,\ \mathsf{F} \mapsto Y[\mathsf{T} \mapsto Z]])) = m(R_a(X_2[Rep])),$$

so we find by (Aux3) that $m((X \mathbin{\tilde{\vee}} Y) \mathbin{\tilde{\wedge}} Z) = m((\rightharpoonup X \mathbin{\tilde{\wedge}} (Y \mathbin{\tilde{\wedge}} Z)) \mathbin{\tilde{\vee}} (X \mathbin{\tilde{\wedge}} Z))$. ■

A.3 A proof of Theorem 4.2

Theorem 4.2: EqMSCL $\vdash$ EqFSCL.

Proof: With help of the theorem prover *Prover9* (McCune, 2008). We derive the EqFSCL-axioms in a particular order to obtain useful intermediate results. Recall that $(n)'$ represents the dual of equation (n).

Axiom (F3). First derive

$$\mathsf{T} \mathbin{\tilde{\vee}} x \overset{\text{(Tand)}}{=} \mathsf{T} \mathbin{\tilde{\wedge}} (\mathsf{T} \mathbin{\tilde{\vee}} x) \overset{(Abs)}{=} \mathsf{T}. \tag{A1}$$

Hence,

$$\begin{aligned}x &= (\mathsf{T} \mathbin{\tilde{\vee}} \ \mathsf{T}) \mathbin{\tilde{\wedge}} x \quad \text{by (Tand), (A1)} \\ &= (\mathsf{F} \mathbin{\tilde{\wedge}} (\mathsf{T} \mathbin{\tilde{\wedge}} x)) \mathbin{\tilde{\vee}} (\mathsf{T} \mathbin{\tilde{\wedge}} x) \quad \text{by (Mem), (Neg)} \\ &= (\mathsf{F} \mathbin{\tilde{\wedge}} x) \mathbin{\tilde{\vee}} x, \quad \text{by (Tand)}\end{aligned} \tag{A2}$$

and

$$\neg(\mathsf{F} \mathbin{\tilde{\wedge}} \neg x) = \neg(\neg \mathsf{T} \mathbin{\tilde{\wedge}} \neg x) \quad \text{by (Neg)}$$

$$= \mathsf{T} \mathbin{\mathring{\vee}} x \quad \text{by (Or)}$$
$$= \mathsf{T}. \quad \text{by (A1)} \tag{A3}$$

Hence, $\neg(\mathsf{F} \mathbin{\mathring{\wedge}} x) \overset{(A2)}{=} \neg(\mathsf{F} \mathbin{\mathring{\wedge}} ((\mathsf{F} \mathbin{\mathring{\wedge}} x) \mathbin{\mathring{\vee}} x)) \overset{(Or)}{=} \neg(\mathsf{F} \mathbin{\mathring{\wedge}} \neg(\neg(\mathsf{F} \mathbin{\mathring{\wedge}} x) \mathbin{\mathring{\wedge}} \neg x)) \overset{(A3)}{=} \mathsf{T}$, and thus

$$\begin{aligned} z &= (\mathsf{T} \mathbin{\mathring{\vee}} y) \mathbin{\mathring{\wedge}} z \quad \text{by (Tand), (A1)} \\ &= (\mathsf{F} \mathbin{\mathring{\wedge}} (y \mathbin{\mathring{\wedge}} z)) \mathbin{\mathring{\vee}} (\mathsf{T} \mathbin{\mathring{\wedge}} z) \quad \text{by (Mem), (Neg)} \\ &= \neg(\neg(\mathsf{F} \mathbin{\mathring{\wedge}} (y \mathbin{\mathring{\wedge}} z)) \mathbin{\mathring{\wedge}} \neg z) \quad \text{by (Tand), (Or)} \\ &= \neg(\mathsf{T} \mathbin{\mathring{\wedge}} \neg z) \quad \text{by } \neg(\mathsf{F} \mathbin{\mathring{\wedge}} x) = \mathsf{T} \\ &= \neg\neg z. \quad \text{by (Tand)} \end{aligned} \tag{F3}$$

Intermediate result 1 – Duality. By axioms (Neg), (Or), (F3), the duality principle holds.

Axiom (F6). $\mathsf{F} \mathbin{\mathring{\wedge}} x = \mathsf{F}$ by (A1)′.

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

$$z = \mathsf{T} \mathbin{\mathring{\wedge}} z \overset{(\text{Tand})'}{=} (\mathsf{F} \mathbin{\mathring{\vee}} \mathsf{T}) \mathbin{\mathring{\wedge}} z \overset{(Mem)}{=} (\mathsf{T} \mathbin{\mathring{\wedge}} (\mathsf{T} \mathbin{\mathring{\wedge}} z)) \mathbin{\mathring{\vee}} (\mathsf{F} \mathbin{\mathring{\wedge}} z) \overset{(\text{Tand}),(F6)}{=} z \mathbin{\mathring{\vee}} \mathsf{F}. \tag{F5}$$

Intermediate result 2 – Idempotence. By axiom (F5), $x = x \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\vee}} \mathsf{F}) \overset{(Abs)}{=} x \mathbin{\mathring{\wedge}} x$.

Axiom (F8). We derive the dual equation. We write 'Idemp' for idempotence and first derive

$$\begin{aligned} x \mathbin{\mathring{\vee}} \mathsf{T} &= (x \mathbin{\mathring{\vee}} \mathsf{T}) \mathbin{\mathring{\wedge}} \mathsf{T} \quad \text{by (F5)}' \\ &= (\neg x \mathbin{\mathring{\wedge}} (\mathsf{T} \mathbin{\mathring{\wedge}} \mathsf{T})) \mathbin{\mathring{\vee}} (x \mathbin{\mathring{\wedge}} \mathsf{T}) \quad \text{by (Mem)} \\ &= \neg x \mathbin{\mathring{\vee}} x, \quad \text{by (F5)}' \end{aligned} \tag{A4}$$

and

$$\neg x \mathbin{\mathring{\vee}} \mathsf{T} = x \mathbin{\mathring{\vee}} \neg x. \quad \text{by (A4), (F3)} \tag{A5}$$

Hence

$$\begin{aligned} x \mathbin{\mathring{\vee}} \mathsf{T} &= (\neg x \mathbin{\mathring{\vee}} x) \mathbin{\mathring{\wedge}} \mathsf{T} \quad \text{by (A4), (F5)}' \\ &= (x \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\wedge}} \mathsf{T})) \mathbin{\mathring{\vee}} (\neg x \mathbin{\mathring{\wedge}} \mathsf{T}) \quad \text{by (Mem)} \\ &= x \mathbin{\mathring{\vee}} \neg x \quad \text{by (F5)}', \text{Idemp} \\ &= \neg x \mathbin{\mathring{\vee}} \mathsf{T}. \quad \text{by (A5)} \end{aligned} \tag{F8′}$$

Intermediate result 3 – four auxiliary results.

$$\begin{aligned} x \mathbin{\mathring{\wedge}} y &= (x \mathbin{\mathring{\vee}} \mathsf{F}) \mathbin{\mathring{\wedge}} y \quad \text{by (F5)} \\ &= (\neg x \mathbin{\mathring{\wedge}} (\mathsf{F} \mathbin{\mathring{\wedge}} y)) \mathbin{\mathring{\vee}} (x \mathbin{\mathring{\wedge}} y) \quad \text{by (Mem)} \\ &= (x \mathbin{\mathring{\wedge}} \mathsf{F}) \mathbin{\mathring{\vee}} (x \mathbin{\mathring{\wedge}} y). \quad \text{by (F6), (F8)} \end{aligned} \tag{Ar1}$$

$$\begin{aligned} x \mathbin{\mathring{\vee}} y &= (x \mathbin{\mathring{\vee}} y) \mathbin{\mathring{\wedge}} \mathsf{T} \quad \text{by (F5)}' \\ &= (\neg x \mathbin{\mathring{\wedge}} y) \mathbin{\mathring{\vee}} x. \quad \text{by (Mem), (F5)}' \end{aligned} \tag{Ar2}$$

$$\begin{aligned} x \mathbin{\mathring{\vee}} y &= (x \mathbin{\mathring{\vee}} \mathsf{T}) \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\vee}} y) \quad \text{by (Ar1)}' \\ &= (\neg x \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\vee}} y)) \mathbin{\mathring{\vee}} (x \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\vee}} y)) \quad \text{by (Mem), (Tand)} \\ &= (\neg x \mathbin{\mathring{\wedge}} (x \mathbin{\mathring{\vee}} y)) \mathbin{\mathring{\vee}} x \quad \text{by (Abs)} \\ &= x \mathbin{\mathring{\vee}} (x \mathbin{\mathring{\vee}} y). \quad \text{by (Ar2)} \end{aligned} \tag{Ar3}$$

$$\begin{aligned} x \mathbin{\mathring{\vee}} y &= (\neg x \mathbin{\mathring{\wedge}} y) \mathbin{\mathring{\vee}} x \quad \text{by (Ar2)} \\ &= (\neg x \mathbin{\mathring{\wedge}} (\neg x \mathbin{\mathring{\wedge}} y)) \mathbin{\mathring{\vee}} x \quad \text{by (Ar3)}' \\ &= x \mathbin{\mathring{\vee}} (\neg x \mathbin{\mathring{\wedge}} y). \quad \text{by (Ar2)} \end{aligned} \tag{Ar4}$$

Axiom (F9). First derive

$$\begin{aligned}(x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} \mathsf{F} &= (\neg x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} \mathsf{F}) \quad \text{by (Mem), (Tand)}\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} \mathsf{F}) \quad \text{by (F8)}\\ &= x \mathbin{\barwedge} \mathsf{F}. \quad \text{by Idemp}\end{aligned} \tag{A6}$$

Hence,

$$\begin{aligned}(x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} y &= ((x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} ((x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} y) \quad \text{by (Ar1)}\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} ((x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} y) \quad \text{by (A6)}\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} (\neg(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y) \quad \text{by (F8)}'\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} y. \quad \text{by (Ar4)}\end{aligned} \tag{F9}$$

Intermediate result 4 – three more auxiliary results.

$$(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y = x \mathbin{\barwedge} \mathsf{F} \tag{Ar5}$$

$$x \mathbin{\barwedge} (y \mathbin{\barwedge} x) = x \mathbin{\barwedge} y \tag{Ar6}$$

$$(x \mathbin{\barwedge} y) \mathbin{\barwedge} x = x \mathbin{\barwedge} y \tag{Ar7}$$

First derive

$$\begin{aligned}(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} \mathsf{F} &= \neg(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} \mathsf{F} \quad \text{by (F8)}\\ &= (\neg x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} \mathsf{F}\\ &= \neg x \mathbin{\barwedge} \mathsf{F} \quad \text{by (A6)}\\ &= x \mathbin{\barwedge} \mathsf{F}, \quad \text{by (F8)}\end{aligned} \tag{A7}$$

hence

$$\begin{aligned}(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y) \quad \text{by (Ar3)}'\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} (((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y)) \quad \text{by (Ar1)}\\ &= (x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} y)) \quad \text{by (A7)}\\ &= x \mathbin{\barwedge} \mathsf{F}. \quad \text{by (Abs)}\end{aligned} \tag{Ar5}$$

$$\begin{aligned}x \mathbin{\barwedge} (y \mathbin{\barwedge} x) &= (x \mathbin{\barwedge} x) \mathbin{\barwedge} (y \mathbin{\barwedge} x) \quad \text{by Idemp}\\ &= ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} x) \mathbin{\barwedge} (y \mathbin{\barwedge} x) \quad \text{by (Ar1), Idemp}\\ &= (\neg(x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} (x \mathbin{\barwedge} (y \mathbin{\barwedge} x))) \mathbin{\overset{\circ}{\vee}} ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\barwedge} (y \mathbin{\barwedge} x)) \quad \text{by (Mem)}\\ &= ((\neg x \mathbin{\overset{\circ}{\vee}} \mathsf{T}) \mathbin{\barwedge} (x \mathbin{\barwedge} (y \mathbin{\barwedge} x))) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} \mathsf{F}) \quad \text{by (Ar5)}\\ &= ((x \mathbin{\barwedge} \mathsf{F}) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} (y \mathbin{\barwedge} x))) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} \mathsf{F}) \quad \text{by (F9), (F8)}\\ &= (x \mathbin{\barwedge} (y \mathbin{\barwedge} x)) \mathbin{\overset{\circ}{\vee}} (x \mathbin{\barwedge} \mathsf{F}) \quad \text{by (Ar1)}\\ &= (x \mathbin{\barwedge} (y \mathbin{\barwedge} x)) \mathbin{\overset{\circ}{\vee}} (\neg x \mathbin{\barwedge} x) \quad \text{by (A4)}'\\ &= (\neg x \mathbin{\overset{\circ}{\vee}} y) \mathbin{\barwedge} x \quad \text{by (Mem)}\\ &= x \mathbin{\barwedge} y. \quad \text{by (Ar2)}'\end{aligned} \tag{Ar6}$$

$$\begin{aligned}(x \mathbin{\barwedge} y) \mathbin{\barwedge} x &= (x \mathbin{\barwedge} y) \mathbin{\barwedge} (x \mathbin{\barwedge} (x \mathbin{\barwedge} y)) \quad \text{by (Ar6)}\\ &= (x \mathbin{\barwedge} y) \mathbin{\barwedge} (x \mathbin{\barwedge} y) \quad \text{by (Ar3)}'\\ &= x \mathbin{\barwedge} y. \quad \text{by Idemp}\end{aligned} \tag{Ar7}$$

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

$$(x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z = (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) \quad (A8)$$

$$(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (y \mathbin{\curlywedge} z) = (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (y \mathbin{\curlywedge} z) \quad (A9)$$

$$\neg x \mathbin{\curlyvee} (y \mathbin{\curlywedge} z) = \neg x \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) \quad (A15)$$

and derive associativity of $\curlywedge$ as follows:

$$\begin{aligned}
(x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z &= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) && \text{by (A8)} \\
&= (x \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) && \text{by (A9)} \\
&= x \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) && \text{by (Abs)}' \\
&= (\neg x \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z)) \mathbin{\curlywedge} x && \text{by (Ar2)}' \\
&= (\neg x \mathbin{\curlyvee} (y \mathbin{\curlywedge} z)) \mathbin{\curlywedge} x && \text{by (A15)} \\
&= x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z). && \text{by (Ar2)}' \qquad (F7)
\end{aligned}$$

We derive the above auxiliary results in order:

$$\begin{aligned}
(x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z) &= ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} \mathsf{F}) && \text{by (Ar7)}' \\
&= ((x \mathbin{\curlyvee} \mathsf{T}) \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} \mathsf{F}) && \text{by (F9)} \\
&= ((\neg x \mathbin{\curlyvee} \mathsf{T}) \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} \mathsf{F}) && \text{by (F8)}' \\
&= (\neg(x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlywedge} ((x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z)) \mathbin{\curlyvee} ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlywedge} z) && \text{by (Ar5)} \\
&= ((x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (x \mathbin{\curlywedge} y)) \mathbin{\curlywedge} z && \text{by (Mem)} \\
&= (x \mathbin{\curlywedge} y) \mathbin{\curlywedge} z. && \text{by (Ar1)} \qquad (A8)
\end{aligned}$$

$$\begin{aligned}
(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (y \mathbin{\curlywedge} z) &= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} (y \mathbin{\curlywedge} z))) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) && \text{by (Mem)} \\
&= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (y \mathbin{\curlywedge} z)) && \text{by (Ar3)}' \\
&= (x \mathbin{\curlyvee} \mathsf{T}) \mathbin{\curlywedge} (y \mathbin{\curlywedge} z) && \text{by (Mem)} \\
&= (x \mathbin{\curlywedge} \mathsf{F}) \mathbin{\curlyvee} (y \mathbin{\curlywedge} z), && \text{by (F9)} \qquad (A9)
\end{aligned}$$

The following auxiliary results lead to (A15):

$$\begin{aligned}
(x \mathbin{\curlyvee} y) \mathbin{\curlywedge} (x \mathbin{\curlyvee} z) &= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} (x \mathbin{\curlyvee} z))) \mathbin{\curlyvee} (x \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)) && \text{by (Mem)} \\
&= (\neg x \mathbin{\curlywedge} (y \mathbin{\curlywedge} (x \mathbin{\curlyvee} z))) \mathbin{\curlyvee} x && \text{by (Abs)} \\
&= x \mathbin{\curlyvee} (y \mathbin{\curlywedge} (x \mathbin{\curlyvee} z)). && \text{by (Ar2)} \qquad (A10)
\end{aligned}$$

$$\begin{aligned}
x \mathbin{\curlyvee} (\neg x \mathbin{\curlyvee} y) &= (\neg x \mathbin{\curlywedge} (\neg x \mathbin{\curlyvee} y)) \mathbin{\curlyvee} x && \text{by (Ar2)} \\
&= \neg x \mathbin{\curlyvee} x && \text{by (Abs)} \\
&= (\neg x \mathbin{\curlywedge} \neg x) \mathbin{\curlyvee} x && \text{by Idemp} \\
&= x \mathbin{\curlyvee} \neg x. && \text{by (Ar2)} \qquad (11)
\end{aligned}$$

$$\begin{aligned}
x \mathbin{\curlyvee} ((x \mathbin{\curlywedge} z) \mathbin{\curlyvee} y) &= x \mathbin{\curlyvee} ((\neg x \mathbin{\curlyvee} (z \mathbin{\curlyvee} y)) \mathbin{\curlywedge} (x \mathbin{\curlyvee} y)) && \text{by (Mem)}' \\
&= (x \mathbin{\curlyvee} (\neg x \mathbin{\curlyvee} (z \mathbin{\curlyvee} y))) \mathbin{\curlywedge} (x \mathbin{\curlyvee} y) && \text{by (A10)} \\
&= (x \mathbin{\curlyvee} \neg x) \mathbin{\curlywedge} (x \mathbin{\curlyvee} y) && \text{by (A11)} \\
&= x \mathbin{\curlyvee} (\neg x \mathbin{\curlywedge} (x \mathbin{\curlyvee} y)) && \text{by (A10)} \\
&= x \mathbin{\curlyvee} (x \mathbin{\curlyvee} y) && \text{by (Ar4)} \\
&= x \mathbin{\curlyvee} y. && \text{by (Ar3)} \qquad (A12)
\end{aligned}$$

$$x \mathbin{\curlyvee} y = x \mathbin{\curlyvee} ((x \mathbin{\curlywedge} z) \mathbin{\curlyvee} y) \quad \text{by (A12)}$$

$$
\begin{aligned}
&= x \mathbin{\barwedge} ((x \mathbin{\barwedge} z) \mathbin{\barvee} (y \mathbin{\barvee} (x \mathbin{\barwedge} z))) \quad \text{by (Ar6)}' \\
&= x \mathbin{\barwedge} (y \mathbin{\barvee} (x \mathbin{\barwedge} z)). \quad \text{by (A12)} \qquad \text{(A13)} \\
x \mathbin{\barwedge} (y \mathbin{\barwedge} z) &= x \mathbin{\barwedge} (\neg x \mathbin{\barwedge} (y \mathbin{\barwedge} z)) \quad \text{by (Ar4)} \\
&= x \mathbin{\barwedge} ((\neg x \mathbin{\barwedge} (y \mathbin{\barwedge} z)) \mathbin{\barvee} (x \mathbin{\barwedge} z)) \quad \text{by (A13)} \\
&= x \mathbin{\barwedge} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z). \quad \text{by (Mem)} \qquad \text{(A14)} \\
\neg x \mathbin{\barwedge} (y \mathbin{\barwedge} z) &= \neg x \mathbin{\barwedge} ((\neg x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by (A14)} \\
&= \neg x \mathbin{\barwedge} ((\neg x \mathbin{\barvee} (x \mathbin{\barwedge} y)) \mathbin{\barwedge} z) \quad \text{by (Ar4)} \\
&= \neg x \mathbin{\barwedge} ((x \mathbin{\barwedge} y) \mathbin{\barwedge} z). \quad \text{by (A14)} \qquad \text{(A15)}
\end{aligned}
$$

We write 'Assoc' for (repeated) applications of associativity of $\barwedge$ and $\barvee$.

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

$$(x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} z) = (\neg x \mathbin{\barvee} y) \mathbin{\barwedge} (x \mathbin{\barvee} z) \qquad \text{(M1)}$$

$$(x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} z) = (\neg x \mathbin{\barwedge} z) \mathbin{\barvee} (x \mathbin{\barwedge} y) \qquad \text{(M2)}$$

Equation (M1).

$$
\begin{aligned}
(x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} z) &= (x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} (x \mathbin{\barvee} z)) \quad \text{by (Ar4)}' \\
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} (x \mathbin{\barvee} z))) \mathbin{\barvee} (\neg x \mathbin{\barwedge} (x \mathbin{\barvee} z)) \quad \text{by (A13)}' \\
&= (\neg x \mathbin{\barvee} y) \mathbin{\barwedge} (x \mathbin{\barvee} z). \quad \text{by (Mem)} \qquad \text{(M1)}
\end{aligned}
$$

Equation (M2). First derive

$$
\begin{aligned}
(x \mathbin{\barvee} y) \mathbin{\barwedge} z &= (x \mathbin{\barvee} y) \mathbin{\barwedge} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by (Ar3)}' \\
&= (x \mathbin{\barvee} (x \mathbin{\barvee} y)) \mathbin{\barwedge} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by (Ar3)} \\
&= (x \mathbin{\barwedge} \mathrm{F}) \mathbin{\barvee} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z), \quad \text{by (A9)} \qquad \text{(A16)} \\
(x \mathbin{\barwedge} \mathrm{F}) \mathbin{\barvee} y &= (\neg x \mathbin{\barwedge} \mathrm{F}) \mathbin{\barvee} y \quad \text{by (F8)} \\
&= (\neg x \mathbin{\barvee} \mathrm{T}) \mathbin{\barwedge} y \quad \text{by (F9)} \\
&= (x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} y) \quad \text{by (Mem), (Tand)} \\
&= (\neg x \mathbin{\barvee} y) \mathbin{\barwedge} (x \mathbin{\barvee} y) \quad \text{by (M1)} \\
&= (\neg x \mathbin{\barvee} (y \mathbin{\barvee} y)) \mathbin{\barwedge} (x \mathbin{\barvee} y) \quad \text{by Idemp} \\
&= (x \mathbin{\barwedge} y) \mathbin{\barvee} y, \quad \text{by (Mem)}' \qquad \text{(A17)} \\
(x \mathbin{\barvee} y) \mathbin{\barwedge} z &= (x \mathbin{\barwedge} \mathrm{F}) \mathbin{\barvee} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by (A16)} \\
&= (x \mathbin{\barwedge} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z)) \mathbin{\barvee} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by (A17)} \\
&= ((x \mathbin{\barwedge} (x \mathbin{\barvee} y)) \mathbin{\barwedge} z) \mathbin{\barvee} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z) \quad \text{by Assoc} \\
&= (x \mathbin{\barwedge} z) \mathbin{\barvee} ((x \mathbin{\barvee} y) \mathbin{\barwedge} z). \quad \text{by (Abs)} \qquad \text{(A18)}
\end{aligned}
$$

Hence,

$$
\begin{aligned}
(x \mathbin{\barwedge} y) \mathbin{\barvee} (\neg x \mathbin{\barwedge} z) &= (\neg x \mathbin{\barvee} y) \mathbin{\barwedge} (x \mathbin{\barvee} z) \quad \text{by (M1)} \\
&= (\neg x \mathbin{\barvee} y) \mathbin{\barwedge} ((x \mathbin{\barvee} z) \mathbin{\barwedge} (\neg x \mathbin{\barvee} y)) \quad \text{by (Ar6)} \\
&= (\neg x \mathbin{\barvee} (x \mathbin{\barwedge} y)) \mathbin{\barwedge} ((x \mathbin{\barvee} z) \mathbin{\barwedge} (\neg x \mathbin{\barvee} y)) \quad \text{by (Ar4)} \\
&= (\neg x \mathbin{\barvee} (x \mathbin{\barwedge} y)) \mathbin{\barwedge} ((\neg x \mathbin{\barwedge} z) \mathbin{\barvee} (x \mathbin{\barwedge} y)) \quad \text{by (M1)}' \\
&= (\neg x \mathbin{\barwedge} z) \mathbin{\barvee} (x \mathbin{\barwedge} y). \quad \text{by (A18)}' \qquad \text{(M2)}
\end{aligned}
$$

Axiom (F10). First derive

$$\begin{aligned}(x \veebar y) \wedge z &= (\neg x \wedge (y \wedge z)) \veebar (x \wedge z) \quad \text{by (Mem)}\\ &= (x \veebar (y \wedge z)) \wedge (\neg x \veebar z). \quad \text{by (M1), (F3)}\end{aligned} \tag{A19}$$

Hence,

$$\begin{aligned}(x \wedge y) \veebar (z \wedge \mathrm{F}) &= (x \wedge (y \veebar (z \wedge \mathrm{F}))) \veebar (\neg x \wedge (z \wedge \mathrm{F})) \quad \text{by (A19)}'\\ &= (x \veebar (z \wedge \mathrm{F})) \wedge (\neg x \veebar (y \veebar (z \wedge \mathrm{F}))) \quad \text{by (M1), (M2)}\\ &= (x \veebar [(z \wedge \mathrm{F}) \wedge (y \veebar (z \wedge \mathrm{F}))])\\ &\quad \wedge (\neg x \veebar (y \veebar (z \wedge \mathrm{F}))) \quad \text{by (Ar5)}\\ &= (x \veebar (z \wedge \mathrm{F})) \wedge (y \veebar (z \wedge \mathrm{F})). \quad \text{by (A19)}\end{aligned} \tag{F10}$$

■

A.4 A proof of Theorem 4.4

Theorem 4.4: *The following equations are derivable from* EqMSCL*, where* (LD) *abbreviates left-distributivity.*

$$x \wedge (y \veebar z) = (x \wedge y) \veebar (x \wedge z), \tag{LD}$$

$$((x \wedge y) \veebar (\neg x \wedge z)) \wedge u = (x \wedge (y \wedge u)) \veebar (\neg x \wedge (z \wedge u)). \tag{M3}$$

Proof: With help of the theorem prover *Prover9* (McCune, 2008).

Equation (LD). First derive

$$\begin{aligned}x \veebar (y \veebar z) &= (x \veebar y) \veebar z \quad \text{by Assoc}\\ &= ((x \veebar y) \veebar x) \veebar z \quad \text{by (Ar7)}'\\ &= x \veebar (y \veebar (x \veebar z)), \quad \text{by Assoc}\end{aligned} \tag{A20}$$

$$\begin{aligned}\neg x \veebar (y \veebar (x \wedge z)) &= \neg x \veebar (y \veebar (\neg x \veebar (x \wedge z))) \quad \text{by (A20)}\\ &= \neg x \veebar (y \veebar (\neg x \veebar z)) \quad \text{by (Ar4)}\\ &= \neg x \veebar (y \veebar z). \quad \text{by (A20)}\end{aligned} \tag{A21}$$

Hence,

$$\begin{aligned}x \wedge (y \veebar z) &= (\neg x \veebar (y \veebar z)) \wedge x \quad \text{by (Ar2)}'\\ &= (\neg x \veebar (y \veebar z)) \wedge (x \veebar (x \wedge z)) \quad \text{by (Abs)}'\\ &= (\neg x \veebar (y \veebar (x \wedge z))) \wedge (x \veebar (x \wedge z)) \quad \text{by (A21)}\\ &= (x \wedge y) \veebar (x \wedge z). \quad \text{by (Mem)}'\end{aligned} \tag{LD}$$

Equation (M3). First derive

$$\begin{aligned}x \wedge (y \wedge ((x \veebar z) \wedge u)) &= (x \wedge (y \wedge (x \veebar z))) \wedge u \quad \text{by Assoc}\\ &= (x \wedge y) \wedge u \quad \text{by (A13)}'\\ &= x \wedge (y \wedge u). \quad \text{by Assoc}\end{aligned} \tag{A22}$$

Hence,

$$\begin{aligned}((x \wedge y) \veebar (\neg x \wedge z)) \wedge u &= ((\neg x \veebar y) \wedge (x \veebar z)) \wedge u \quad \text{by (M1)}\\ &= (\neg x \veebar y) \wedge ((x \veebar z) \wedge u) \quad \text{by Assoc}\end{aligned}$$

$$
\begin{aligned}
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} ((x \mathbin{\barvee} z) \mathbin{\barwedge} u))) \mathbin{\barvee} \\
&\qquad (\neg x \mathbin{\barwedge} ((x \mathbin{\barvee} z) \mathbin{\barwedge} u)) \quad \text{by (Mem)} \\
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} u)) \mathbin{\barvee} (\neg x \mathbin{\barwedge} ((x \mathbin{\barvee} z) \mathbin{\barwedge} u)) \quad \text{by (A22)} \\
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} u)) \mathbin{\barvee} ((\neg x \mathbin{\barwedge} (x \mathbin{\barvee} z)) \mathbin{\barwedge} u) \quad \text{by Assoc} \\
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} u)) \mathbin{\barvee} ((\neg x \mathbin{\barwedge} z) \mathbin{\barwedge} u) \quad \text{by (Ar4)}' \\
&= (x \mathbin{\barwedge} (y \mathbin{\barwedge} u)) \mathbin{\barvee} (\neg x \mathbin{\barwedge} (z \mathbin{\barwedge} u)). \quad \text{by Assoc} \qquad \text{(M3)}
\end{aligned}
$$

■

A.5 A proof of Lemma 4.7

Lemma 4.7: *For all $a \in A$, $P \in \mathcal{S}_A$, and $Q \in$ MSNF,*

(1) EqMSCL $\vdash a \mathbin{\barwedge} (P \mathbin{\barwedge} Q) = a \mathbin{\barwedge} (P \mathbin{\barwedge} T_a(Q))$,
(2) EqMSCL $\vdash \neg a \mathbin{\barwedge} (P \mathbin{\barwedge} Q) = \neg a \mathbin{\barwedge} (P \mathbin{\barwedge} F_a(Q))$,
(3) EqMSCL $\vdash a \mathbin{\barwedge} (P \mathbin{\barvee} Q) = a \mathbin{\barwedge} (P \mathbin{\barvee} T_a(Q))$,
(4) EqMSCL $\vdash \neg a \mathbin{\barwedge} (P \mathbin{\barvee} Q) = \neg a \mathbin{\barwedge} (P \mathbin{\barvee} F_a(Q))$.

Proof: Statement 1 follows by induction on the structure of Q. If $Q \in \{\mathsf{T}, \mathsf{F}\}$ this is trivial. For the induction step there are two cases: if $Q = (a \mathbin{\barwedge} Q_1) \mathbin{\barvee} (\neg a \mathbin{\barwedge} Q_2)$, then derive from EqMSCL (tacitly using Assoc)

$$
\begin{aligned}
a \mathbin{\barwedge} (P \mathbin{\barwedge} Q) &= (a \mathbin{\barwedge} (P \mathbin{\barwedge} (a \mathbin{\barwedge} Q_1))) \mathbin{\barvee} (a \mathbin{\barwedge} (P \mathbin{\barwedge} (\neg a \mathbin{\barwedge} Q_2))) \quad \text{by (LD)} \\
&= (a \mathbin{\barwedge} (P \mathbin{\barwedge} Q_1)) \mathbin{\barvee} (a \mathbin{\barwedge} (P \mathbin{\barwedge} \mathsf{F})) \quad \text{by (C2), (A21)}', \text{(F6)} \\
&= (a \mathbin{\barwedge} P) \mathbin{\barwedge} (Q_1 \mathbin{\barvee} \mathsf{F}) \quad \text{by (LD)} \\
&= a \mathbin{\barwedge} (P \mathbin{\barwedge} T_a(Q)), \quad \text{by (F5)}
\end{aligned}
$$

and if $Q = (b \mathbin{\barwedge} Q_1) \mathbin{\barvee} (\neg b \mathbin{\barwedge} Q_2)$, then derive from EqMSCL (tacitly using Assoc)

$$
\begin{aligned}
&a \mathbin{\barwedge} (P \mathbin{\barwedge} Q) \\
&\quad = (a \mathbin{\barwedge} (P \mathbin{\barwedge} (b \mathbin{\barwedge} Q_1))) \mathbin{\barvee} (a \mathbin{\barwedge} (P \mathbin{\barwedge} (\neg b \mathbin{\barwedge} Q_2))) \quad \text{by (LD)} \\
&\quad = (a \mathbin{\barwedge} (P \mathbin{\barwedge} (b \mathbin{\barwedge} (a \mathbin{\barwedge} (P \mathbin{\barwedge} Q_1))))) \mathbin{\barvee} (a \mathbin{\barwedge} (P \mathbin{\barwedge} (\neg b \mathbin{\barwedge} (a \mathbin{\barwedge} (P \mathbin{\barwedge} Q_2))))) \quad \text{by (C2)} \\
&\quad = (a \mathbin{\barwedge} (P \mathbin{\barwedge} (b \mathbin{\barwedge} (a \mathbin{\barwedge} (P \mathbin{\barwedge} T_a(Q_1)))))) \mathbin{\barvee} \\
&\qquad (a \mathbin{\barwedge} (P \mathbin{\barwedge} (\neg b \mathbin{\barwedge} (a \mathbin{\barwedge} (P \mathbin{\barwedge} T_a(Q_2)))))) \quad \text{by IH} \\
&\quad = (a \mathbin{\barwedge} (P \mathbin{\barwedge} (b \mathbin{\barwedge} T_a(Q_1)))) \mathbin{\barvee} (a \mathbin{\barwedge} (P \mathbin{\barwedge} (\neg b \mathbin{\barwedge} (T_a(Q_2))))) \quad \text{by (C2)} \\
&\quad = (a \mathbin{\barwedge} P) \mathbin{\barwedge} ((b \mathbin{\barwedge} T_a(Q_1)) \mathbin{\barvee} (\neg b \mathbin{\barwedge} (T_a(Q_2)))) \quad \text{by (LD)} \\
&\quad = a \mathbin{\barwedge} (P \mathbin{\barwedge} T_a(Q)).
\end{aligned}
$$

The remaining statements follow in a similar way. ■

A.6 Continuation of the proof of Lemma 6.4

Lemma 6.4: *For all $s, t \in \mathbb{T}_{\Sigma_{\mathrm{CP}}(A),\mathcal{X}}$,* $\mathrm{CP}_{mem} \vdash s = t \Rightarrow$ EqMSCL $\vdash g(s) = g(t)$.

Proof: The cases for axioms (CP4) and (CPmem).

Axiom (CP4). We use Equations (M1) and (M2) (see Fact 4.3), and (M3) and (LD) (see Theorem 4.4), and we write 'Assoc' for associativity.

$$g(x \lhd (y \lhd z \rhd u) \rhd v)$$

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

and

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

Axiom (CPmem). As argued in Section 5, it is sufficient to derive axiom (CPmem1), that is,

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

This is straightforward, a detailed proof is provided in Appendix A.6. Derive

$$
\begin{aligned}
g((w \lhd y \rhd (z \lhd x \rhd u)) \lhd x \rhd v) &= (x \owedge g(w \lhd y \rhd (z \lhd x \rhd u))) \ovee (\neg x \owedge v) \quad \text{by (2)}\\
&= (x \owedge M) \ovee (\neg x \owedge v),\\
g((w \lhd y \rhd z) \lhd x \rhd v) &= (x \owedge g(w \lhd y \rhd z)) \ovee (\neg x \owedge v) \quad \text{by (2)}\\
&= (x \owedge N) \ovee (\neg x \owedge v),
\end{aligned}
$$

so it suffices to derive $x \owedge M = x \owedge N$. We use one auxiliary result and we write $(n)'$ for the dual version of equation (n), and 'Idemp' for idempotence.

$$
\begin{aligned}
x \owedge \mathsf{F} &= (x \owedge \mathsf{F}) \owedge y \quad \text{by (F6), Assoc}\\
&= (x \owedge \neg x) \owedge y. \quad \text{by (C1), (F3)}
\end{aligned}
\tag{A23}
$$

Hence,

$$
\begin{aligned}
x \owedge M &= x \owedge g(w \lhd y \rhd (z \lhd x \rhd u))\\
&= x \owedge ((y \owedge w) \ovee (\neg y \owedge ((x \owedge z) \ovee (\neg x \owedge u))))\\
&= x \owedge ((\neg y \ovee w) \owedge (y \ovee ((x \owedge z) \ovee (\neg x \owedge u)))) \quad \text{by (M1)}\\
&= x \owedge ((y \ovee ((x \owedge z) \ovee (\neg x \owedge u))) \owedge (\neg y \ovee w)) \quad \text{by (M2)}\\
&= x \owedge ((y \ovee ((\neg x \owedge u) \ovee (x \owedge z))) \owedge (\neg y \ovee w)) \quad \text{by (M2)}\\
&= [x \owedge (y \ovee ((\neg x \owedge u) \ovee (x \owedge z)))] \owedge (\neg y \ovee w) \quad \text{by Assoc}\\
&= [(x \owedge y) \ovee (x \owedge ((\neg x \owedge u) \ovee (x \owedge z)))] \owedge (\neg y \ovee w) \quad \text{by (LD)}
\end{aligned}
$$

$$
\begin{aligned}
&= [(x \mathbin{⍟\wedge} y) \mathbin{⍟\vee} ((x \mathbin{⍟\wedge} (\neg x \mathbin{⍟\wedge} u)) \mathbin{⍟\vee} (x \mathbin{⍟\wedge} (x \mathbin{⍟\wedge} z)))] \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w) && \text{by (LD)}\\
&= [(x \mathbin{⍟\wedge} y) \mathbin{⍟\vee} (((x \mathbin{⍟\wedge} \neg x) \mathbin{⍟\wedge} u) \mathbin{⍟\vee} (x \mathbin{⍟\wedge} z))] \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w) && \text{by Assoc, Idemp}\\
&= [(x \mathbin{⍟\wedge} y) \mathbin{⍟\vee} ((x \mathbin{⍟\wedge} \mathsf{F}) \mathbin{⍟\vee} (x \mathbin{⍟\wedge} z))] \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w) && \text{by (A23)}\\
&= [(x \mathbin{⍟\wedge} y) \mathbin{⍟\vee} (x \mathbin{⍟\wedge} (\mathsf{F} \mathbin{⍟\vee} z))] \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w) && \text{by (LD)}\\
&= (x \mathbin{⍟\wedge} (y \mathbin{⍟\vee} z)) \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w) && \text{by (Tand)}', \text{(LD)}\\
&= x \mathbin{⍟\wedge} ((y \mathbin{⍟\vee} z) \mathbin{⍟\wedge} (\neg y \mathbin{⍟\vee} w)) && \text{by Assoc}\\
&= x \mathbin{⍟\wedge} g(w \mathbin{\triangleleft} y \mathbin{\triangleright} z) && \text{by (3)}\\
&= x \mathbin{⍟\wedge} N.
\end{aligned}
$$

■

A.7 A proof of Theorem 7.12

Theorem 7.12: *The axioms of* EqMSCL$^{\mathsf{U}}$ *are independent.*

Proof: All independence models were found with the tool *Mace4* (McCune, 2008). In each model $\mathbb{M}$ defined below, $[\![\mathsf{F}]\!]^{\mathbb{M}} = 0$ and $[\![\mathsf{T}]\!]^{\mathbb{M}} = 1$.

Independence of axiom (Neg). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(\text{Neg})\}$ with domain $\{0, 1, 2\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 0$ that refutes $\mathsf{F} = \neg\,\mathsf{T}$ is the following:

$\neg$	
0	0
1	2
2	1

$⍟\wedge$	0	1	2
0	0	0	0
1	0	1	2
2	2	2	2

$⍟\vee$	0	1	2
0	0	0	0
1	1	1	1
2	0	1	2

Independence of axiom (Or). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(\text{Or})\}$ with domain $\{0, 1\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 0$ that refutes $\mathsf{T} \mathbin{⍟\vee} \mathsf{F} = \neg(\neg\,\mathsf{T}$
leftand$\neg\,\mathsf{F})$ is the following:

$\neg$	
0	0
1	0

$⍟\wedge$	0	1
0	0	1
1	0	1

$⍟\vee$	0	1
0	0	0
1	1	1

Independence of axiom (Tand). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(\text{Tand})\}$ with domain $\{0, 1\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 0$ that refutes $\mathsf{T} \mathbin{⍟\wedge} \mathsf{F} = \mathsf{F}$ is the following:

$\neg$	
0	0
1	0

$⍟\wedge$	0	1
0	0	0
1	1	0

$⍟\vee$	0	1
0	0	0
1	0	0

Independence of axiom (Abs). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(\mathit{Abs})\}$ with domain $\{0, 1\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 0$ that refutes $\mathsf{T} \mathbin{⍟\wedge} (\mathsf{T} \mathbin{⍟\vee} \mathsf{F}) = \mathsf{T}$ is the following:

$\neg$	
0	0
1	0

$⍟\wedge$	0	1
0	0	0
1	0	1

$⍟\vee$	0	1
0	0	0
1	0	0

Independence of axiom (Mem). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(\mathit{Mem})\}$ with domain $\{0, 1, 2\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 2$ that refutes $(\mathsf{F} \mathbin{⍟\vee} \mathsf{T}) \mathbin{⍟\wedge} \mathsf{U} = (\neg\,\mathsf{F} \mathbin{⍟\wedge} (\mathsf{T} \mathbin{⍟\wedge} \mathsf{U})) \mathbin{⍟\vee} (\mathsf{F} \mathbin{⍟\wedge} \mathsf{U})$ is the following:

$\neg$	
0	1
1	0
2	2

$⍟\wedge$	0	1	2
0	0	0	0
1	0	1	2
2	2	1	2

$⍟\vee$	0	1	2
0	0	1	2
1	1	1	1
2	0	2	2

Independence of axiom (4). A model $\mathbb{M}$ for EqMSCL$^{\mathsf{U}} \setminus \{(4)\}$ with domain $\{0,1\}$ and $[\![\mathsf{U}]\!]^{\mathbb{M}} = 0$ that refutes $\neg\mathsf{U} = \mathsf{U}$ is the following:

$\neg$	
0	1
1	0

$\barwedge$	0	1
0	0	0
1	0	1

$\barvee$	0	1
0	0	1
1	1	1

■

A.8 CP$^{\mathsf{U}}$ and evaluation trees

Finally, we prove Proposition 7.13: this is Theorem A.6 below. This text (excluding footnotes) comes from Bergstra and Ponse (2017), although the signature has now been extended with the constant U, and CP to CP$^{\mathsf{U}}$ with the axiom (CP-U), that is, $x \lhd \mathsf{U} \rhd y = \mathsf{U}$.

Let $\mathcal{C}_A^{\mathsf{U}}$ be the set of closed terms over $\Sigma_{\mathrm{CP}^{\mathsf{U}}}(A)$.

Definition A.1: *Basic forms over A* are defined by the following grammar

$$t ::= \mathsf{T} \mid \mathsf{F} \mid \mathsf{U} \mid t \lhd a \rhd t \quad \text{for } a \in A.$$

We write BF_A^* for the set of basic forms over A.

The following lemmas exploit the structure of basic forms.[8]

Lemma A.2: *For each $P \in \mathcal{C}_A^{\mathsf{U}}$ there exists $Q \in BF_A^*$ such that* CP$^{\mathsf{U}} \vdash P = Q$.

Proof: First we establish an auxiliary result: if P, Q, R are basic forms, then there is a basic form S such that CP $\vdash P \lhd Q \rhd R = S$. This follows by structural induction on Q.

The lemma's statement follows by structural induction on P.

The base cases $P \in \{\mathsf{T}, \mathsf{F}, \mathsf{U}, a \mid a \in A\}$ are trivial, and if $P = P_1 \lhd P_2 \rhd P_3$ there exist by induction basic forms Q_i such that CP $\vdash P_i = Q_i$, hence CP $\vdash P_1 \lhd P_2 \rhd P_3 = Q_1 \lhd Q_2 \rhd Q_3$. Now apply the auxiliary result. ■

Lemma A.3: *For all basic forms P and Q, $se^{\mathsf{U}}(P) = se^{\mathsf{U}}(Q)$ implies $P = Q$.*

Proof: By structural induction on P. The base cases $P \in \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ are trivial. If $P = P_1 \lhd a \rhd P_2$, then $Q \notin \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ and $Q \neq Q_1 \lhd b \rhd Q_2$ if $b \neq a$, so $Q = Q_1 \lhd a \rhd Q_2$ and $se^{\mathsf{U}}(P_i) = se^{\mathsf{U}}(Q_i)$. By induction we find $P_i = Q_i$, and hence $P = Q$. ■

Definition A.4: *Free valuation congruence*, notation $=_{se^{\mathsf{U}}}$, is defined on $\mathcal{C}_A^{\mathsf{U}}$ as follows:

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

Lemma A.5: *Free valuation congruence is a congruence relation.*

Proof: Let $P, Q, R \in \mathcal{C}_A^{\mathsf{U}}$ and assume $P =_{se^{\mathsf{U}}} P'$, thus $se^{\mathsf{U}}(P) = se^{\mathsf{U}}(P')$. Then $se^{\mathsf{U}}(P \lhd Q \rhd R) = se^{\mathsf{U}}(Q)[\mathsf{T} \mapsto se^{\mathsf{U}}(P), \mathsf{F} \mapsto se^{\mathsf{U}}(R)] = se^{\mathsf{U}}(Q)[\mathsf{T} \mapsto se^{\mathsf{U}}(P'), \mathsf{F} \mapsto se^{\mathsf{U}}(R)] = se^{\mathsf{U}}(P' \lhd Q \rhd R)$, and thus $P \lhd Q \rhd R =_{se^{\mathsf{U}}} P' \lhd Q \rhd R$. The two remaining cases can be proved in a similar way. ■

Theorem A.6 (Completeness of CP$^{\mathsf{U}}$ for closed terms): *For all $P, Q \in \mathcal{C}_A^{\mathsf{U}}$,*

$$\mathrm{CP}^{\mathsf{U}} \vdash P = Q \iff P =_{se^{\mathsf{U}}} Q.$$

Proof: ($\Rightarrow$)[9] By Lemma A.5, $=_{se^{\cup}}$ is a congruence relation and it easily follows that closed instances of CP^{U}-axioms are valid. In the case of axiom (CP4) this follows from

$$
\begin{aligned}
&se^{\mathsf{U}}(P \lhd (Q \lhd R \rhd S) \rhd V)\\
&\quad = se^{\mathsf{U}}(Q \lhd R \rhd S)[\mathsf{T} \mapsto se^{\mathsf{U}}(P), \mathsf{F} \mapsto se^{\mathsf{U}}(V)]\\
&\quad = \big(se^{\mathsf{U}}(R)[\mathsf{T} \mapsto se^{\mathsf{U}}(Q), \mathsf{F} \mapsto se^{\mathsf{U}}(S)]\big)\,[\mathsf{T} \mapsto se^{\mathsf{U}}(P), \mathsf{F} \mapsto se^{\mathsf{U}}(V)]\\
&\quad = se^{\mathsf{U}}(R)[\mathsf{T} \mapsto se^{\mathsf{U}}(Q)[\mathsf{T} \mapsto se^{\mathsf{U}}(P), \mathsf{F} \mapsto se^{\mathsf{U}}(V)], \mathsf{F} \mapsto se^{\mathsf{U}}(S)[\mathsf{T} \mapsto se^{\mathsf{U}}(P), \mathsf{F} \mapsto se^{\mathsf{U}}(V)]]\\
&\quad = se^{\mathsf{U}}(R)[\mathsf{T} \mapsto se^{\mathsf{U}}(P \lhd Q \rhd V), \mathsf{F} \mapsto se^{\mathsf{U}}(P \lhd S \rhd V)]\\
&\quad = se^{\mathsf{U}}((P \lhd Q \rhd V) \lhd R \rhd (P \lhd S \rhd V)).
\end{aligned}
$$

($\Leftarrow$) Let $P =_{se^{\cup}} Q$. According to Lemma A.2 there exist basic forms P' and Q' such that $\mathrm{CP}^{\mathsf{U}} \vdash P = P'$ and $\mathrm{CP}^{\mathsf{U}} \vdash Q = Q'$. By ($\Rightarrow$) we find $P =_{se^{\cup}} P'$ and $Q =_{se^{\cup}} Q'$, and because $=_{se^{\cup}}$ is a congruence, $P' =_{se^{\cup}} Q'$. By Lemma A.3, $P' = Q'$. Hence, $\mathrm{CP}^{\mathsf{U}} \vdash P = P' = Q' = Q$. ■