Automated Reasoning

Christoph Weidenbach

Max Planck Institute for Informatics

October 17, 2024

Automated Reasoning

Given a specification of a system, develop technology

logics,
calculi,
algorithms,
implementations,

to automatically execute the specification and to automatically prove properties of the specification.

Concept

Slides: Definitions, Lemmas, Theorems, . . .

Written: Examples, Proofs, . . .

Speech: Motivate, Explain, . . .

Script: Slides, partially Blackboard . . .

Exams: able to calculate $\rightarrow$ pass
understand $\rightarrow$ (very) good grade

Orderings

1.4.1 Definition (Orderings)

A *(partial) ordering* $\succeq$ (or simply ordering) on a set M, denoted $(M, \succeq)$, is a reflexive, antisymmetric, and transitive binary relation on M.

It is a *total ordering* if it also satisfies the totality property.

A *strict (partial) ordering* $\succ$ is a transitive and irreflexive binary relation on M.

A strict ordering is *well-founded*, if there is no infinite descending chain $m_0 \succ m_1 \succ m_2 \succ \ldots$ where $m_i \in M$.

1.4.3 Definition (Minimal and Smallest Elements)

Given a strict ordering $(M, \succ)$, an element $m \in M$ is called *minimal*, if there is no element $m' \in M$ so that $m \succ m'$.

An element $m \in M$ is called *smallest*, if $m' \succ m$ for all $m' \in M$ different from m.

Multisets

Given a set M, a *multiset* S over M is a mapping $S\colon M \to \mathbb{N}$, where S specifies the number of occurrences of elements m of the base set M within the multiset S. I use the standard set notations $\in$, $\subset$, $\subseteq$, $\cup$, $\cap$ with the analogous meaning for multisets, for example $(S_1 \cup S_2)(m) = S_1(m) + S_2(m)$.

A multiset S over a set M is *finite* if $\{m \in M \mid S(m) > 0\}$ is finite. For the purpose of this lecture I only consider finite multisets.

1.4.5 Definition (Lexicographic and Multiset Ordering Extensions)

Let $(M_1, \succ_1)$ and $(M_2, \succ_2)$ be two strict orderings.

Their *lexicographic combination* $\succ_{\text{lex}} = (\succ_1, \succ_2)$ on $M_1 \times M_2$ is defined as $(m_1, m_2) \succ (m_1', m_2')$ iff $m_1 \succ_1 m_1'$ or $m_1 = m_1'$ and $m_2 \succ_2 m_2'$.

Let $(M, \succ)$ be a strict ordering.

The *multiset extension* $\succ_{\text{mul}}$ to multisets over M is defined by $S_1 \succ_{\text{mul}} S_2$ iff $S_1 \neq S_2$ and $\forall m \in M\, [S_2(m) > S_1(m) \rightarrow \exists m' \in M\, (m' \succ m \wedge S_1(m') > S_2(m'))]$.

1.4.7 Proposition (Properties of $\succ_{lex}$, $\succ_{mul}$)

Let $(M, \succ)$, $(M_1, \succ_1)$, and $(M_2, \succ_2)$ be orderings. Then

1. $\succ_{lex}$ is an ordering on $M_1 \times M_2$.
2. if $(M_1, \succ_1)$, $(M_2, \succ_2)$ are well-founded so is $\succ_{lex}$.
3. if $(M_1, \succ_1)$, $(M_2, \succ_2)$ are total so is $\succ_{lex}$.
4. $\succ_{mul}$ is an ordering on multisets over M.
5. if $(M, \succ)$ is well-founded so is $\succ_{mul}$.
6. if $(M, \succ)$ is total so is $\succ_{mul}$.

Please recall that multisets are finite.

Induction

Theorem (Noetherian Induction)

Let $(M, \succ)$ be a well-founded ordering, and let Q be a predicate over elements of M. If for all $m \in M$ the implication

if $Q(m')$, for all $m' \in M$ so that $m \succ m'$,	(induction hypothesis)
then $Q(m)$.	(induction step)

is satisfied, then the property $Q(m)$ holds for all $m \in M$.

Abstract Rewrite Systems

1.6.1 Definition (Rewrite System)

A *rewrite system* is a pair $(M, \rightarrow)$, where M is a non-empty set and $\rightarrow \subseteq M \times M$ is a binary relation on M.

$\rightarrow^0$	$= \{ (a, a) \mid a \in M \}$	*identity*
$\rightarrow^{i+1}$	$= \rightarrow^i \circ \rightarrow$	*$i + 1$-fold composition*
$\rightarrow^+$	$= \bigcup_{i>0} \rightarrow^i$	*transitive closure*
$\rightarrow^*$	$= \bigcup_{i\geq 0} \rightarrow^i \; = \; \rightarrow^+ \cup \rightarrow^0$	*reflexive transitive closure*
$\rightarrow^=$	$= \rightarrow \cup \rightarrow^0$	*reflexive closure*
$\rightarrow^{-1}$	$= \leftarrow \; = \{ (b, c) \mid c \rightarrow b \}$	*inverse*
$\leftrightarrow$	$= \rightarrow \cup \leftarrow$	*symmetric closure*
$\leftrightarrow^+$	$= (\leftrightarrow)^+$	*transitive symmetric closure*
$\leftrightarrow^*$	$= (\leftrightarrow)^*$	*refl. trans. symmetric closure*

1.6.2 Definition (Reducible)

Let $(M, \rightarrow)$ be a rewrite system. An element $a \in M$ is *reducible*, if there is a $b \in M$ such that $a \rightarrow b$.

An element $a \in M$ is *in normal form (irreducible)*, if it is not reducible.

An element $c \in M$ is a *normal form* of b, if $b \rightarrow^* c$ and c is in normal form, denoted by $c = b{\downarrow}$.

Two elements b and c are *joinable*, if there is an a so that $b \rightarrow^* a \;{}^*\!\!\leftarrow c$, denoted by $b \downarrow c$.

1.6.3 Definition (Properties of $\rightarrow$)

A relation $\rightarrow$ is called

Church-Rosser	if $b \leftrightarrow^* c$ implies $b \downarrow c$
confluent	if $b \;{}^*\!\!\leftarrow a \rightarrow^* c$ implies $b \downarrow c$
locally confluent	if $b \leftarrow a \rightarrow c$ implies $b \downarrow c$
terminating	if there is no infinite descending chain $b_0 \rightarrow b_1 \rightarrow b_2 \ldots$
normalizing	if every $b \in A$ has a normal form
convergent	if it is confluent and terminating

1.6.4 Lemma (Termination vs. Normalization)

If $\rightarrow$ is terminating, then it is normalizing.

1.6.5 Theorem (Church-Rosser vs. Confluence)

The following properties are equivalent for any $(M, \rightarrow)$:

(i) $\rightarrow$ has the Church-Rosser property.

(ii) $\rightarrow$ is confluent.

1.6.6 Lemma (Newman's Lemma)

Let $(M, \rightarrow)$ be a terminating rewrite system. Then the following properties are equivalent:

(i) $\rightarrow$ is confluent

(ii) $\rightarrow$ is locally confluent

LA Equations Rewrite System

M is the set of all LA equations sets N over $\mathbb{Q}$

$\doteq$ includes normalizing the equation

Eliminate $\{x \doteq s, x \doteq t\} \uplus N \Rightarrow_{\text{LAE}} \{x \doteq s, x \doteq t, s \doteq t\} \cup N$
provided $s \neq t$, and $s \doteq t \notin N$

Fail $\{q_1 \doteq q_2\} \uplus N \Rightarrow_{\text{LAE}} \emptyset$
provided $q_1, q_2 \in \mathbb{Q}$, $q_1 \neq q_2$

LAE Redundancy

Subsume $\{s \doteq t, s' \doteq t'\} \uplus N \;\Rightarrow_{\text{LAE}}\; \{s \doteq t\} \cup N$

provided $s \doteq t$ and $qs' \doteq qt'$ are identical for some $q \in \mathbb{Q}$

Rewrite Systems on Logics: Calculi

	Validity	Satisfiability
Sound	If the calculus derives a proof of validity for the formula, it is valid.	If the calculus derives satisfiability of the formula, it has a model.
Complete	If the formula is valid, a proof of validity is derivable by the calculus.	If the formula has a model, the calculus derives satisfiability.
Strongly Complete	For any validity proof of the formula, there is a derivation in the calculus producing this proof.	For any model of the formula, there is a derivation in the calculus producing this model.

Propositional Logic: Syntax

2.1.1 Definition (Propositional Formula)

The set PROP(Σ) of *propositional formulas* over a signature Σ, is inductively defined by:

PROP(Σ)	Comment
$\bot$	connective $\bot$ denotes "false"
$\top$	connective $\top$ denotes "true"
P	for any propositional variable $P \in \Sigma$
$(\neg\phi)$	connective $\neg$ denotes "negation"
$(\phi \land \psi)$	connective $\land$ denotes "conjunction"
$(\phi \lor \psi)$	connective $\lor$ denotes "disjunction"
$(\phi \rightarrow \psi)$	connective $\rightarrow$ denotes "implication"
$(\phi \leftrightarrow \psi)$	connective $\leftrightarrow$ denotes "equivalence"

where $\phi, \psi \in$ PROP(Σ).

Propositional Logic: Semantics

2.2.1 Definition ((Partial) Valuation)

A Σ-*valuation* is a map

$$\mathcal{A} : \Sigma \to \{0, 1\}.$$

where $\{0, 1\}$ is the set of *truth values*. A *partial* Σ-*valuation* is a map $\mathcal{A}' : \Sigma' \to \{0, 1\}$ where $\Sigma' \subseteq \Sigma$.

2.2.2 Definition (Semantics)

A Σ-valuation $\mathcal{A}$ is inductively extended from propositional variables to propositional formulas $\phi, \psi \in$ PROP(Σ) by

$$
\begin{aligned}
\mathcal{A}(\bot) &:= 0\\
\mathcal{A}(\top) &:= 1\\
\mathcal{A}(\neg\phi) &:= 1 - \mathcal{A}(\phi)\\
\mathcal{A}(\phi \land \psi) &:= \min(\{\mathcal{A}(\phi), \mathcal{A}(\psi)\})\\
\mathcal{A}(\phi \lor \psi) &:= \max(\{\mathcal{A}(\phi), \mathcal{A}(\psi)\})\\
\mathcal{A}(\phi \rightarrow \psi) &:= \max(\{1 - \mathcal{A}(\phi),\ \mathcal{A}(\psi)\})\\
\mathcal{A}(\phi \leftrightarrow \psi) &:= \text{if } \mathcal{A}(\phi) = \mathcal{A}(\psi) \text{ then 1 else 0}
\end{aligned}
$$

If $\mathcal{A}(\phi) = 1$ for some Σ-valuation $\mathcal{A}$ of a formula ϕ then ϕ is *satisfiable* and we write $\mathcal{A} \models \phi$. In this case $\mathcal{A}$ is a *model* of ϕ.

If $\mathcal{A}(\phi) = 1$ for all Σ-valuations $\mathcal{A}$ of a formula ϕ then ϕ is *valid* and we write $\models \phi$.

If there is no Σ-valuation $\mathcal{A}$ for a formula ϕ where $\mathcal{A}(\phi) = 1$ we say ϕ is *unsatisfiable*.

A formula ϕ *entails* ψ, written $\phi \models \psi$, if for all Σ-valuations $\mathcal{A}$ whenever $\mathcal{A} \models \phi$ then $\mathcal{A} \models \psi$.

Propositional Logic: Operations

2.1.2 Definition (Atom, Literal, Clause)

A propositional variable P is called an *atom*. It is also called a *(positive) literal* and its negation $\neg P$ is called a *(negative) literal*.

The functions comp and atom map a literal to its complement, or atom, respectively: if $\text{comp}(\neg P) = P$ and $\text{comp}(P) = \neg P$, $\text{atom}(\neg P) = P$ and $\text{atom}(P) = P$ for all $P \in \Sigma$. Literals are denoted by letters L, K. Two literals P and $\neg P$ are called *complementary*.

A disjunction of literals $L_1 \vee \ldots \vee L_n$ is called a *clause*. A clause is identified with the multiset of its literals.

2.1.3 Definition (Position)

A *position* is a word over $\mathbb{N}$. The set of positions of a formula ϕ is inductively defined by

$$
\begin{array}{rcl}
\mathrm{pos}(\phi) & := & \{\epsilon\} \text{ if } \phi \in \{\top, \bot\} \text{ or } \phi \in \Sigma \\
\mathrm{pos}(\neg\phi) & := & \{\epsilon\} \cup \{1p \mid p \in \mathrm{pos}(\phi)\} \\
\mathrm{pos}(\phi \circ \psi) & := & \{\epsilon\} \cup \{1p \mid p \in pos(\phi)\} \cup \{2p \mid p \in \mathrm{pos}(\psi)\}
\end{array}
$$

where $\circ \in \{\land, \lor, \rightarrow, \leftrightarrow\}$.

The prefix order $\leq$ on positions is defined by $p \leq q$ if there is some p' such that $pp' = q$. Note that the prefix order is partial, e.g., the positions 12 and 21 are not comparable, they are "parallel", see below.

The relation $<$ is the strict part of $\leq$, i.e., $p < q$ if $p \leq q$ but not $q \leq p$.

The relation $\parallel$ denotes incomparable, also called parallel positions, i.e., $p \parallel q$ if neither $p \leq q$, nor $q \leq p$.

A position p is *above* q if $p \leq q$, p is *strictly above* q if $p < q$, and p and q are *parallel* if $p \parallel q$.

The *size* of a formula ϕ is given by the cardinality of $\text{pos}(\phi)$: $|\phi| := |\text{pos}(\phi)|$.

The *subformula* of ϕ at position $p \in \text{pos}(\phi)$ is inductively defined by $\phi|_\epsilon := \phi$, $\neg\phi|_{1p} := \phi|_p$, and $(\phi_1 \circ \phi_2)|_{ip} := \phi_i|_p$ where $i \in \{1, 2\}$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Finally, the *replacement* of a subformula at position $p \in \text{pos}(\phi)$ by a formula ψ is inductively defined by $\phi[\psi]_\epsilon := \psi$, $(\neg\phi)[\psi]_{1p} := \neg\phi[\psi]_p$, and $(\phi_1 \circ \phi_2)[\psi]_{1p} := (\phi_1[\psi]_p \circ \phi_2)$, $(\phi_1 \circ \phi_2)[\psi]_{2p} := (\phi_1 \circ \phi_2[\psi]_p)$, where $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

2.1.5 Definition (Polarity)

The *polarity* of the subformula $\phi|_p$ of ϕ at position $p \in \mathrm{pos}(\phi)$ is inductively defined by

$$
\begin{array}{rcl}
\mathrm{pol}(\phi, \epsilon) & := & 1 \\
\mathrm{pol}(\neg\phi, 1p) & := & -\mathrm{pol}(\phi, p) \\
\mathrm{pol}(\phi_1 \circ \phi_2, ip) & := & \mathrm{pol}(\phi_i, p) \quad \text{if } \circ \in \{\wedge, \vee\}, i \in \{1,2\} \\
\mathrm{pol}(\phi_1 \rightarrow \phi_2, 1p) & := & -\mathrm{pol}(\phi_1, p) \\
\mathrm{pol}(\phi_1 \rightarrow \phi_2, 2p) & := & \mathrm{pol}(\phi_2, p) \\
\mathrm{pol}(\phi_1 \leftrightarrow \phi_2, ip) & := & 0 \quad \text{if } i \in \{1,2\}
\end{array}
$$

Valuations can be nicely represented by sets or sequences of literals that do not contain complementary literals nor duplicates.

If $\mathcal{A}$ is a (partial) valuation of domain Σ then it can be represented by the set
$\{P \mid P \in \Sigma \text{ and } \mathcal{A}(P) = 1\} \cup \{\neg P \mid P \in \Sigma \text{ and } \mathcal{A}(P) = 0\}$.

Another, equivalent representation are *Herbrand* interpretations that are sets of positive literals, where all atoms not contained in an Herbrand interpretation are false. If $\mathcal{A}$ is a total valuation of domain Σ then it corresponds to the Herbrand interpretation
$\{P \mid P \in \Sigma \text{ and } \mathcal{A}(P) = 1\}$.

2.2.4 Theorem (Deduction Theorem)

$\phi \models \psi$ iff $\models \phi \rightarrow \psi$

2.2.6 Lemma (Formula Replacement)

Let ϕ be a propositional formula containing a subformula ψ at position p, i.e., $\phi|_p = \psi$. Furthermore, assume $\models \psi \leftrightarrow \chi$. Then $\models \phi \leftrightarrow \phi[\chi]_p$.

Normal Forms

Definition (CNF, DNF)

A formula is in *conjunctive normal form (CNF)* or *clause normal form* if it is a conjunction of disjunctions of literals, or in other words, a conjunction of clauses.

A formula is in *disjunctive normal form (DNF)*, if it is a disjunction of conjunctions of literals.

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy:

(i) a formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals P and $\neg P$,

(ii) conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals P and $\neg P$

Basic CNF Transformation

ElimEquiv	$\chi[(\phi \leftrightarrow \psi)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)]_p$
ElimImp	$\chi[(\phi \rightarrow \psi)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[(\neg\phi \vee \psi)]_p$
PushNeg1	$\chi[\neg(\phi \vee \psi)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[(\neg\phi \wedge \neg\psi)]_p$
PushNeg2	$\chi[\neg(\phi \wedge \psi)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[(\neg\phi \vee \neg\psi)]_p$
PushNeg3	$\chi[\neg\neg\phi]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\phi]_p$
PushDisj	$\chi[(\phi_1 \wedge \phi_2) \vee \psi]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[(\phi_1 \vee \psi) \wedge (\phi_2 \vee \psi)]_p$
ElimTB1	$\chi[(\phi \wedge \top)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\phi]_p$
ElimTB2	$\chi[(\phi \wedge \bot)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\bot]_p$
ElimTB3	$\chi[(\phi \vee \top)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\top]_p$
ElimTB4	$\chi[(\phi \vee \bot)]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\phi]_p$
ElimTB5	$\chi[\neg\bot]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\top]_p$
ElimTB6	$\chi[\neg\top]_p$	$\Rightarrow_{\text{BCNF}}$	$\chi[\bot]_p$

Basic CNF Algorithm

Algorithm: 2 bcnf(ϕ)

Input : A propositional formula ϕ.

Output : A propositional formula ψ equivalent to ϕ in CNF.

whilerule *(***ElimEquiv**(ϕ)*)* **do** ;

;

whilerule *(***ElimImp**(ϕ)*)* **do** ;

;

whilerule *(***ElimTB1**(ϕ),. . .,**ElimTB6**(ϕ)*)* **do** ;

;

whilerule *(***PushNeg1**(ϕ),. . .,**PushNeg3**(ϕ)*)* **do** ;

;

whilerule *(***PushDisj**(ϕ)*)* **do** ;

;

return ϕ;

Advanced CNF Algorithm

For the formula

$$P_1 \leftrightarrow (P_2 \leftrightarrow (P_3 \leftrightarrow (\ldots (P_{n-1} \leftrightarrow P_n) \ldots)))$$

the basic CNF algorithm generates a CNF with 2^{n-1} clauses.

2.5.4 Proposition (Renaming Variables)

Let P be a propositional variable not occurring in $\psi[\phi]_p$.

1. If $\text{pol}(\psi, p) = 1$, then $\psi[\phi]_p$ is satisfiable if and only if $\psi[P]_p \wedge (P \rightarrow \phi)$ is satisfiable.
2. If $\text{pol}(\psi, p) = -1$, then $\psi[\phi]_p$ is satisfiable if and only if $\psi[P]_p \wedge (\phi \rightarrow P)$ is satisfiable.
3. If $\text{pol}(\psi, p) = 0$, then $\psi[\phi]_p$ is satisfiable if and only if $\psi[P]_p \wedge (P \leftrightarrow \phi)$ is satisfiable.

Renaming

SimpleRenaming $\phi \Rightarrow_{\text{SimpRen}} \phi[P_1]_{p_1}[P_2]_{p_2}\ldots[P_n]_{p_n} \wedge \text{def}(\phi, p_1, P_1) \wedge \ldots \wedge \text{def}(\phi[P_1]_{p_1}[P_2]_{p_2}\ldots[P_{n-1}]_{p_{n-1}}, p_n, P_n)$
provided $\{p_1, \ldots, p_n\} \subset \text{pos}(\phi)$ and for all $i, i+j$ either $p_i \parallel p_{i+j}$ or $p_i > p_{i+j}$ and the P_i are different and new to ϕ

Simple choice: choose $\{p_1, \ldots, p_n\}$ to be all non-literal and non-negation positions of ϕ.

Renaming Definition

$$\mathrm{def}(\psi, p, P) := \begin{cases} (P \rightarrow \psi|_p) & \text{if } \mathrm{pol}(\psi, p) = 1 \\ (\psi|_p \rightarrow P) & \text{if } \mathrm{pol}(\psi, p) = -1 \\ (P \leftrightarrow \psi|_p) & \text{if } \mathrm{pol}(\psi, p) = 0 \end{cases}$$

Obvious Positions

A smaller set of positions from ϕ, called *obvious positions*, is still preventing the explosion and given by the rules:

(i) p is an obvious position if $\phi|_p$ is an equivalence and there is a position $q < p$ such that $\phi|_q$ is either an equivalence or disjunctive in ϕ or

(ii) pq is an obvious position if $\phi|_{pq}$ is a conjunctive formula in ϕ, $\phi|_p$ is a disjunctive formula in ϕ, $q \neq \epsilon$, and for all positions r with $p < r < pq$ the formula $\phi|_r$ is not a conjunctive formula.

A formula $\phi|_p$ is conjunctive in ϕ if $\phi|_p$ is a conjunction and $\text{pol}(\phi, p) \in \{0, 1\}$ or $\phi|_p$ is a disjunction or implication and $\text{pol}(\phi, p) \in \{0, -1\}$.

Analogously, a formula $\phi|_p$ is disjunctive in ϕ if $\phi|_p$ is a disjunction or implication and $\text{pol}(\phi, p) \in \{0, 1\}$ or $\phi|_p$ is a conjunction and $\text{pol}(\phi, p) \in \{0, -1\}$.

Polarity Dependent Equivalence Elimination

ElimEquiv1 $\chi[(\phi \leftrightarrow \psi)]_p \Rightarrow_{\text{ACNF}} \chi[(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)]_p$
provided $\text{pol}(\chi, p) \in \{0, 1\}$

ElimEquiv2 $\chi[(\phi \leftrightarrow \psi)]_p \Rightarrow_{\text{ACNF}} \chi[(\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)]_p$
provided $\text{pol}(\chi, p) = -1$

Extra $\top, \bot$ Elimination Rules

ElimTB7	$\chi[\phi \rightarrow \bot]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\neg\phi]_p$
ElimTB8	$\chi[\bot \rightarrow \phi]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\top]_p$
ElimTB9	$\chi[\phi \rightarrow \top]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\top]_p$
ElimTB10	$\chi[\top \rightarrow \phi]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\phi]_p$
ElimTB11	$\chi[\phi \leftrightarrow \bot]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\neg\phi]_p$
ElimTB12	$\chi[\phi \leftrightarrow \top]_p$	$\Rightarrow_{\text{ACNF}}$	$\chi[\phi]_p$

where the two rules ElimTB11, ElimTB12 for equivalences are applied with respect to commutativity of $\leftrightarrow$.

Advanced CNF Algorithm

Algorithm: 3 acnf(ϕ)

Input : A formula ϕ.

Output : A formula ψ in CNF satisfiability preserving to ϕ.

```
whilerule (ElimTB1(ϕ),...,ElimTB12(ϕ)) do ;
;
SimpleRenaming(ϕ) on obvious positions;
whilerule (ElimEquiv1(ϕ),ElimEquiv2(ϕ)) do ;
;
whilerule (ElimImp(ϕ)) do ;
;
whilerule (PushNeg1(ϕ),...,PushNeg3(ϕ)) do ;
;
whilerule (PushDisj(ϕ)) do ;
;
return ϕ;
```