Consistent consequences formalized

Myrthe van Delft

August 31, 2016

Abstract

Model checking is the problem of determining if a system, represented by a *model*, satisfies a certain property. *Parameterized Boolean equation systems*, or *PBES* for short can (among others) be used to encode model checking problems. From the solution of a PBES the solution of the encoded model checking problem can be obtained. The solution of a PBES is an assignment of Boolean values to its variables. It is undecidable whether a variable has a specific value in the solution of a PBES. Therefore, syntactic transformation and abstraction techniques are often used to simplify PBES. Proving the soundness of such techniques is a laborious and difficult process. The notion of consistent consequence has been defined to simplify such soundness proofs.

Consistent consequence relations relate the solution of a variable from one PBES, to the solution of a variable from another PBES. Showing that variables are related via a consistent consequence relation is difficult. This is because consistent consequence is a complicated notion based on semantics. To gain greater insight in the notion of consistent consequence, a proof system has been defined to derive a consistent consequence between variables. There was a small error in this proof system. In this master thesis, we have fixed this error, and formalized the theory in the proof assistant Coq to increase the confidence in the results. In this report we present the new proof system and the theory which has been formalized in Coq.

Contents

1 Introduction **1**
- 1.1 Contents of this report . . . 2

2 Definitions of consequence **4**
- 2.1 Propositional formulas and consequences . . . 4
- 2.2 Relations and relativizations . . . 7
- 2.3 BES . . . 14

3 Solutions of consequence **18**
- 3.1 The consistent consequence relation . . . 18
- 3.2 An important theorem . . . 20
- 3.3 On variations of consistent consequences . . . 23

4 A proof system of consequence **25**
- 4.1 Some examples . . . 29

5 Proof system consistency **30**
- 5.1 Soundness . . . 32
- 5.2 Completeness for logical consequence . . . 34
- 5.3 Completeness . . . 43

6 Consequences in Coq **49**
- 6.1 Background . . . 49
- 6.2 The contents of this chapter . . . 50
- 6.3 Propositional formulas . . . 53
- 6.4 Relations on variables . . . 59
- 6.5 Boolean Equation Systems . . . 62
- 6.6 Well-formed BES . . . 72
- 6.7 Proof system for consistent consequences . . . 76
- 6.8 Semantics of BES . . . 81

7 Conclusion **84**
- 7.1 The future: From BES to PBES . . . 85

A Full proof system **86**

B Equivalent semantics **88**

C PBES rules **91**

D The remaining parts of Coq **103**

Chapter 1

Introduction

Model checking refers to the following problem. Does a system, whose behavior is encoded in a *model* (for example a transition system), satisfy some property. For example consider the system of a railroad crossing. A property which we wish to hold for this system, is that whenever a train is coming, the bars close before the train has arrived, to prevent accidents. Furthermore, we wish that cars can cross the crossing when it is safe to do so. Checking if these properties are satisfied are typical examples of model checking problems. A way to do this, is to encode the model checking problem as a *Parameterised Boolean equation system* (PBES for short) [13], and then solve this PBES.

PBESs can be used to encode a variety of verification problems, including (first-order) modal μ-calculus model checking problems [12], behavioral equivalence problems such as bisimilarity, similarity and branching bisimilarity [4] and model checking problems for real-time systems [25]. A PBES is a list of fixpoint equations using variables and predicate formulas, for example:

$$(\mu X(n : Nat) = X(n+1))(\nu Y(k : Nat) = \mathit{even}(k) \vee (Y(k+1) \wedge X(k+2)))$$

Solving this PBES means finding an assignment for the predicate variables in the PBES, such that μX yields the smallest solution and νY yields the largest solution satisfying the equations. By solving a PBES resulting from an encoding of a particular verification problem, an answer to the encoded problem is obtained. Advanced tool suites, such as mCRL2 [6] and CADP [9], rely on the use of PBES for solving their verification problems.

The solution of a PBES is an assignement of Boolean values to its variables. It is undecidable whether a variable has a specific value in the solution of a PBES. However, the problem is decidable for certain fragments of PBES, such as Boolean equation systems (BES for short)[18]. A common technique for solving PBES, is to instantiate the PBES in BES, and solve the resulting BES. However, this transformation process is not necessarily finite, and furthermore it suffers from an exponential blow up similar to the state space explosion problem.

Therefore, abstraction and transformation techniques are often used to simplify PBES. In [23], a method is presented for determining the soundness of such techniques in the setting of BES. This method relies on identifying *consistent correlations* between equation systems, and the decidability of consistent correlation (i.e. determining if the solution of a specific equation is related via consistent correlation to another equation) was addressed for a fragment of BES.

The notion of consistent correlation was designed to simplify the soundness proofs of transformation techniques. Soundness of a transformation can be shown by showing that the equations in the resulting PBES are related via a consistent correlation to the equations from the original PBES. However, a limitation of the consistent correlations is that they can only be used for verifying reflecting, solution preserving transformations, such as removing redundant parameters [19]. (For example, in the example shown before, n is a redundant parameter for the solution of X.)

Abstraction techniques often result in PBESs whose solutions are under- or over-approximations of the original PBES's solution. To accommodate these techniques as well, the notion of consistent consequence has been defined. Consistent consequence lies at the basis of consistent correlation. Intuitively, where consistent correlation relates solutions of variables from PBES via bi-implication, consistent consequence relates solutions of variables from PBES via implication.

Consistent consequence is itself a form of abstraction on PBES. Given two PBES, we say that a variable from one PBES is a consistent consequence of a variable from the other PBES, if there exists a consistent consequence relation which relates these variables. However, showing that there exists such a consistent consequence relation is complex. This is because consistent consequence is a complicated notion based on the semantics of PBES. In [10], the notion of consistent consequence has been studied for BES, and a complete and sound proof system which would allow for syntactically deriving consistent consequences between variables was proposed. However, it was later discovered that this proof system was unsound.

1.1 Contents of this report

In this report, we will show that by adjusting the proof system from [10], we obtain a proof system which *is* sound and complete for deriving consistent consequences. Furthermore, to increase the confidence in the results this has been formalized in Coq. We also formalized some of the theory underlying the consistent consequence relation. In particular, the relation between the solution of variables in BES and notion of consistent consequence was formalized.

Considering that Coq is not a mainstream method for presenting such theories and proofs, in Chapters 2, 3, 4 and 5 we present the theory which has been formalized in Coq, with references to their Coq counterparts (Chapter 6).

In Chapter 2 we present definitions for propositional formulas, logical and relative consequence, relations on propositional variables and some of their properties, as well as BES and their semantics. In Chapter 3 we present the consistent consequence relation, and show the relation between the notion of consistent consequence and the semantics of BES. In Chapter 4, we present a proof system for deriving consistent consequences, and discuss why the version from [10] was unsound and what we did to fix this problem. In Chapter 5, we prove that the proof system is sound and complete for deriving consistent consequences.

In each of these chapters, the most important lemmas will contain references to Chapter 6. Chapter 6 starts with an introduction to Coq, and then continues with a documentation of the Coq files containing the formalization, which can be found in an SVN repository: [7]. It discusses the types and definitions as defined in Coq and the choices that have been made in the formalization, as well as explaining the theorems and lemmas which were proven in Coq.

Chapter 2

Definitions of consequence

In this chapter, we present definitions for Boolean Equation systems, as well as some auxiliary definitions. Furthermore, we will present some examples and useful properties of these definitions.

2.1 Propositional formulas and consequences

We assume some infinite set $\mathcal{X}$ of propositional variables. These variables are typically indicated with a capital X, Y or Z. We will indicate the domain of the Booleans with $\mathbb{B}$. Boolean values will be written as *true* and *false*.

Using propositional variables and some standard connectives, we can define the syntax of propositional formulas.

Definition 2.1.1. Propositional formulas are recursively defined by the following grammar.

$$f ::= \top \mid \bot \mid X \mid f \wedge f \mid f \vee f$$

Propositional formulas will typically be indicated by a lowercase f or g. Given a propositional formula f, the set of variables occurring in f is $\mathcal{V}(f)$. If a propositional formula follows a certain pattern, then we say that the formula is in disjunctive normal form (DNF), or that it is a DNF.

Definition 2.1.2. Let f be a propositional formula. If f is $\top$, $\bot$ or some propositional variable then f is a *singleton.* Furthermore, f is a clause if it is a singleton or a conjunction of clauses. Finally, f is in *disjunctive normal form* (it is a DNF) if it is a clause or a disjunction of DNFs.

Given a propositional formula f in DNF, the set of clauses occurring in disjunctions in f is indicated by $\mathcal{C}(f)$.

We will use the term *environment* to indicate a function which interprets propositional variables, by assigning a Boolean value to every variable. Typically, we will use $\theta : \mathcal{X} \to \mathbb{B}$, or simply θ, to indicate environments. Given two environments, the union of these environments is defined as follows.

Definition 2.1.3. Given two environments θ_1, θ_2, the *union* of θ_1 and θ_2, written $\theta_1 || \theta_2$, is defined as the environment assigning *true* to any variable X iff θ_1 or θ_2 assigns *true* to this variable.

We can use an environment θ to define another environment which differs from θ in (at most) one variable.

Definition 2.1.4. Given an environment θ and propositional variable X, we will write $\theta[X := b]$ for some $b \in \mathbb{B}$ to indicate that θ has been *redefined in X* to b, defined as follows.

$$\theta[X := b](Y) := \begin{cases} b & \text{if } X = Y \\ \theta(Y) & \text{otherwise} \end{cases}$$

By lifting the notion of environment in the natural way, we obtain the semantics of propositional formulas.

Definition 2.1.5. Given an environment $\theta : \mathcal{X} \to \mathbb{B}$, the semantics (or meaning) of a propositional formula f is defined as follows.

$$\begin{aligned}
[\![\top]\!]\theta &= \mathit{true} \\
[\![\bot]\!]\theta &= \mathit{false} \\
[\![X]\!]\theta &= \begin{cases} \mathit{true} & \text{if } \theta(X) = \mathit{true} \\ \mathit{false} & \text{otherwise} \end{cases} \\
[\![f \wedge g]\!]\theta &= \begin{cases} \mathit{true} & \text{if both } [\![f]\!]\theta = \mathit{true} \text{ and } [\![g]\!]\theta = \mathit{true} \\ \mathit{false} & \text{otherwise} \end{cases} \\
[\![f \vee g]\!]\theta &= \begin{cases} \mathit{true} & \text{if } [\![f]\!]\theta = \mathit{true} \text{ or } [\![g]\!]\theta = \mathit{true} \\ \mathit{false} & \text{otherwise} \end{cases}
\end{aligned}$$

A basic concept on the semantics of propositional formulas is the logical implication, which relates propositional formulas based on their semantics. We will refer to this concept as *logical consequence* in this text.

Definition 2.1.6. Given two propositional formulas f, g. If, for all environments θ, we have $[\![f]\!]\theta = \mathit{true}$ implies $[\![g]\!]\theta = \mathit{true}$, then we say that g is a *logical consequence*, or *consequence* of f. We may write $f \Rightarrow g$ to indicate that g is a logical consequence of f.

Given two propositional formulas f, g, if both f is a consequence of g, and g is a consequence of f, then the two are equivalent.

Definition 2.1.7. Given two propositional formulas f, g, if both f is a consequence of g, and g is a consequence of f, then we say that f is *logically equivalent*, or *equivalent*, to g. We may write $f \Leftrightarrow g$ to indicate that f and g are equivalent.

Since we are dealing with positive propositional formulas, we can derive a few interesting properties which would not hold on propositional formulas which allow negation. We end this section with a few of these properties. The first is that, given a clause f not equivalent to $\bot$ and a propositional g, g is a logical consequence of f iff g is assigned *true* to the environment assigning *false* to all variables except for the variables which occur in f.

Lemma 2.1.8. For any clause f not equivalent to $\bot$ and propositional formula g, $f \Rightarrow g$ iff g is *true* under the environment assigning *false* to all variables except for the variables from $\mathcal{V}(f)$.

Proof. Take a clause f which is not equivalent to $\bot$, and let θ_f be the environment assigning *false* to all variables except for the variables from $\mathcal{V}(f)$. Note that $[\![f]\!]\theta = \mathit{true}$.

- $\rightarrow$ Take any propositional formula g such that $f \Rightarrow g$. Then since $[\![f]\!]\theta_f = \mathit{true}$ also $[\![g]\!]\theta_f = \mathit{true}$.

- $\leftarrow$ Take any propositional formula g such that $[\![g]\!]\theta_f = \mathit{true}$. We proceed by induction on the structure of g.

  - $\top$ In this case trivially $f \Rightarrow g$.

  - $\bot$ In this case $[\![g]\!]\theta_f = \mathit{false}$, which contradicts our assumptions.

  - Y In this case, since θ_f only assigns *true* to variables from $\mathcal{V}(f)$, we have that $Y \in \mathcal{V}(f)$. Take any environment θ such that $[\![f]\!]\theta = \mathit{true}$. This is only the case if for all variables $X \in \mathcal{V}(f)$ we have that $\theta(X) = \mathit{true}$, and therefore also $\theta(Y) = \mathit{true}$ and thus $f \Rightarrow g$.

  - $g_1 \wedge g_2$ In this case we have the following induction hypotheses:
    IH$_1$: If $[\![g_1]\!]\theta_f = \mathit{true}$ then $f \Rightarrow g_1$.
    IH$_2$: If $[\![g_2]\!]\theta_f = \mathit{true}$ then $f \Rightarrow g_2$.

    Since $[\![g]\!]\theta_f = \mathit{true}$, both $[\![g_1]\!]\theta_f = \mathit{true}$ and $[\![g_2]\!]\theta_f = \mathit{true}$, and thus from the induction hypotheses both $f \Rightarrow g_1$ and $f \Rightarrow g_2$ hold. Take any environment θ such that $[\![f]\!]\theta = \mathit{true}$. We need to show that also $[\![g]\!]\theta = \mathit{true}$. This is the case if both $[\![g_1]\!]\theta = \mathit{true}$ and $[\![g_2]\!]\theta = \mathit{true}$, which follows from the fact that both $f \Rightarrow g_1$ and $f \Rightarrow g_2$ hold.

  - $g_1 \vee g_2$ In this case we have the following induction hypotheses:
    IH$_1$: If $[\![g_1]\!]\theta_f = \mathit{true}$ then $f \Rightarrow g_1$.
    IH$_2$: If $[\![g_2]\!]\theta_f = \mathit{true}$ then $f \Rightarrow g_2$.

    Since $[\![g]\!]\theta_f = \mathit{true}$, we have that $[\![g_1]\!]\theta_f = \mathit{true}$ or $[\![g_2]\!]\theta_f = \mathit{true}$. The cases are symmetrical, so we will only consider $[\![g_1]\!]\theta_f = \mathit{true}$. In this case from IH$_1$ we have that $f \Rightarrow g_1$. Take any environment θ such that $[\![f]\!]\theta = \mathit{true}$. Then also $[\![g_1]\!] = \mathit{true}$, and thus $[\![g_1 \vee g_2]\!] = \mathit{true}$, therefore $f \Rightarrow g$. □

An interesting consequence of this property is that for a propositional formula $g_1 \vee g_2$ which is a logical consequence of some clause, at least one of g_1 and g_2 is also a logical consequence of this clause.

Lemma 2.1.9. For any clause f and propositional formulas g_1, g_2 such that $f \Rightarrow (g_1 \vee g_2)$, we have that $f \Rightarrow g_1$ or $f \Rightarrow g_2$.

Proof. Take a clause f and propositional formulas g_1, g_2 such that $f \Rightarrow (g_1 \vee g_2)$. If f is equivalent to $\bot$ then trivially $f \Rightarrow g_1$ (and $f \Rightarrow g_2$).

Assume that f is not equivalent to $\bot$, and consider the environment θ_f which assigns *false* to all variables except for the variables from $\mathcal{V}(f)$. Then since $f \Rightarrow (g_1 \vee g_2)$, we have $[\![g_1]\!]\theta_f = \mathit{true}$ or $[\![g_2]\!]\theta_f = \mathit{true}$. Therefore, from Lemma 2.1.8, we have that $f \Rightarrow g_1$ or $f \Rightarrow g_2$. □

If we are dealing with DNFs, then we have an even stronger property; if a DNF g is a logical consequence of a DNF f, then for all clauses in f there exists a clause in g such that the clause from g is a logical consequence of the clause from f.

Lemma 2.1.10. For any DNFs f, g such that $f \Rightarrow g$ we have that, for all clauses $c_f \in \mathcal{C}(f)$, there exists a clause $c_g \in \mathcal{C}(g)$ such that $c_f \Rightarrow c_g$.

Proof. Take a DNF g, we proceed by induction on the structure of g.

$\top, \bot, X, c_g$ Take a DNF f such that $f \Rightarrow g$. In these cases, there is only one clause $c_g \in \mathcal{C}(g)$. Let c_f be a clause from $\mathcal{C}(f)$. Now take any environment θ such that $[\![c_f]\!]\theta = \mathit{true}$. Then, since f is in DNF, also $[\![f]\!]\theta = \mathit{true}$ and thus also $[\![g]\!]\theta = \mathit{true}$. Therefore, $c_f \Rightarrow g$. Since $g = c_g$ also $c_g \in \mathcal{C}(g)$, also $c_f \Rightarrow c_g$.

$g_1 \vee g_2$ In this case we have the following induction hypotheses:
IH$_1$: For all DNF's f' such that $f' \Rightarrow g_1$ then for all clauses $c_f \in \mathcal{C}(f')$ there exists a clause $c_g \in \mathcal{C}(g_1)$ such that $c_f \Rightarrow c_g$.
IH$_2$: For all DNF's f' such that $f' \Rightarrow g_2$ then for all clauses $c_f \in \mathcal{C}(f')$ there exists a clause $c_g \in \mathcal{C}(g_2)$ such that $c_f \Rightarrow c_g$.

Take a DNF f such that $f \Rightarrow g$. Let c_f be a clause from $\mathcal{C}(f)$. Now take any environment θ such that $[\![c_f]\!]\theta = \mathit{true}$. Then, since f is in DNF, also $[\![f]\!]\theta = \mathit{true}$ and thus also $[\![g]\!]\theta = \mathit{true}$. Therefore, $c_f \Rightarrow g$.

Now consider the environment θ_f which assigns *false* to all variables except for the variables from $\mathcal{V}(c_f)$. The, since $c_f \Rightarrow g$, we have that $[\![g_1]\!]\theta_f = \mathit{true}$ or $[\![g_2]\!]\theta_f = \mathit{true}$. The cases are symmetrical, so we will only consider the case $[\![g_1]\!]\theta_f = \mathit{true}$.

If $[\![g_1]\!]\theta_f = \mathit{true}$, then from Lemma 2.1.8 we know that $c_f \Rightarrow g_1$. Since c_f is a clause, c_f is also a DNF, and thus from IH$_1$ we have that there must be a clause $c_g \in \mathcal{C}(g_1)$ such that $c_f \Rightarrow c_g$. Take this clause c_g, then since $c_g \in \mathcal{C}(g_1)$ also $c_g \in \mathcal{C}(g_1 \vee g_2)$, and therefore there exists a clause $c_g \in \mathcal{C}(g_1 \vee g_2)$ such that $c_f \Rightarrow c_g$. $\square$

2.2 Relations and relativizations

We use $\mathfrak{I}$ to denote the identity relation. Furthermore, given a relation $R \subseteq \mathcal{X} \times \mathcal{X}$ on propositional variables, we may write $X \; R \; Y$ instead of $(X, Y) \in R$.

A relation on propositional variables can be lifted to a relation on propositional formulas, using the following notion of consistent environments.

Definition 2.2.1. Given a relation on propositional variables R and an environment θ, we say that θ is *consistent with* R if, for all $(X, Y) \in R$, $\theta(X) = \mathit{true}$ implies $\theta(Y) = \mathit{true}$. The set of all environments consistent with R is Θ_R, and we may write $\theta \in \Theta_R$ if an environment θ is consistent with R.

The environment assigning *true* to all variables is indicated with θ_ν, the environment assigning *false* to all variables is indicated with θ_μ. For any relation R, both θ_ν and θ_μ are consistent with R.

Lemma 2.2.2. For any relation R, both θ_ν and θ_μ are consistent with R.

Proof. Let R be some relation on propositional variables, and take some propositional variables $(X, Y) \in R$. Then, $\theta_\sigma(X) = \theta_\sigma(Y)$ for $\sigma \in \{\mu, \nu\}$, and thus θ_σ is consistent with R for both $\sigma = \mu$ and $\sigma = \nu$. $\square$

Given two environments consistent with R, their union is also consistent.

Lemma 2.2.3. For any relation R and environments θ_1, θ_2. $\theta_1 || \theta_2$ is consistent with R if both θ_1 and θ_2 are consistent with R.

Proof. Let R be some relation on propositional variables, and take environments θ_1, θ_2 consistent with R. Furthermore, take some propositional variables $(X, Y) \in R$ such that $(\theta_1 || \theta_2)(X) = \mathit{true}$. We need to show that $(\theta_1 || \theta_2)(Y) = \mathit{true}$. If $(\theta_1 || \theta_2)(X) = \mathit{true}$ then $\theta_1(X) = \mathit{true}$ or $\theta_2(X) = \mathit{true}$.

The cases are symmetrical, so we will only consider $\theta_1(X) = \mathit{true}$. In this case, since θ_1 is consistent with R, also $\theta_1(Y) = \mathit{true}$. Therefore, from the definition of the union of environments, $(\theta_1 || \theta_2)(Y) = \mathit{true}$. $\square$

The notion of consistency between an environment and a relation can be used to lift relations on propositional variables to relations on propositional formulas, via the following definition.

Definition 2.2.4. Given a relation R on propositional variables and propositional formulas f and g. We say that g is a *consequence of f relative to R* if, for all environments $\theta \in \Theta_R$, $[\![f]\!]\theta = \mathit{true}$ implies $[\![g]\!]\theta = \mathit{true}$. If g is a consequence of f relative to R then we may write $f \overset{R}{\Rightarrow} g$.

Relative consequence is a weaker form of logical consequence, i.e. if we have a relative consequence between propositional formulas, then we also have a logical consequence between these formulas.

Lemma 2.2.5. For any propositional formulas f, g, if $f \Rightarrow g$ then for any relation R on propositional variables, $f \overset{R}{\Rightarrow} g$.

Proof. Take propositional formulas f, g such that $f \Rightarrow g$. Then, for any environment θ we have $[\![f]\!]\theta = \mathit{true}$ implies $[\![g]\!]\theta = \mathit{true}$. Thus this is also the case for all environments consistent with R, and therefore $f \overset{R}{\Rightarrow} g$. $\square$

However, the other direction of the previous lemma is not true. Consider the following examples.

Example 2.2.6. For any propositional variables X and Y, we have $X \wedge Y \Rightarrow X$. Also, obviously, for any relation R on propositional variables, $X \wedge Y \overset{R}{\Rightarrow} X$.

However, $X \vee Y \Rightarrow X$ does not hold. Consider an environment θ with $\theta(Y) = \mathit{true}$ and $\theta(X) = \mathit{false}$. Then $[\![X \vee Y]\!]\theta = \mathit{true}$ but $[\![X]\!]\theta = \mathit{false}$, thus $X \vee Y \Rightarrow X$ does not hold. However, if we consider some relation R on propositional variables such that $(Y, X) \in R$, then any environment consistent with R which assigns *true* to either X or Y, must (also) assign *true* to X (since it is consistent with R), and therefore $X \vee Y \overset{R}{\Rightarrow} X$.

Given a relation R on propositional variables, we will use R^* to indicate the reflexive transitive closure of R, and R^+ to indicate the transitive closure of R. Given a relation R and propositional variable X, we can find the variables Y such that (X, Y) is in the transitive closure of R using the following definition.

Definition 2.2.7. Given a relation R on propositional variables and propositional variables X, Y. We say that Y *is reachable from* X *through* R if there exist variables $X_0, \cdots, X_n$ for some $n > 0$ such that $X_0 = X$, $X_n = Y$ and for all $i < n$ we have $X_i \; R \; X_{i+1}$.

There is a one to one correspondence between membership of a pair of variables in the transitive closure of a relation, and reachability between these variables through this relation.

Lemma 2.2.8. For all relations R on propositional variables and propositional variables X, Y, Y is reachable from X through R if and only if $(X, Y) \in R^+$.

Proof. This follows trivially from the definition of the transitive closure of a relation. $\square$

For any relation R, the set of environments consistent with R is the same as the set of environments consistent with R^+.

Lemma 2.2.9. For all relations R on propositional variables we have $\Theta_R = \Theta_{R^+}$.

Proof. Take some relation R on propositional variables and environment θ. We need to show that $\theta \in \Theta_R$ iff $\theta \in \Theta_{R^+}$.

$\rightarrow$ Assume that $\theta \in \Theta_R$. Furthermore, take some variable X such that $\theta(X) = \textit{true}$. We need to show that for all variables Y such that $(X, Y) \in R^+$, $\theta(Y) = \textit{true}$. If for some variable Y we have $(X, Y) \in R^+$, then there exist $n + 1$ variables $X_0, \cdots, X_n$ such that $X_0 = X$, $X_n = Y$ and $X_i \; R \; X_{i+1}$ for all $i < n$.

We will show that, for all variables $X_0, \cdots, X_n$ such that $X_0 = X$ and $X_i \; R \; X_{i+1}$ for all $i < n$, we have $\theta(X_n) = \textit{true}$. We prove this by induction on n.

$n = 1$ In this case $\theta(X_n) = \textit{true}$ trivially holds.

$n = k + 1$ In this case we have the following induction hypothesis. For all variables $X_0, \cdots, X_k$ such that $X_0 = X$ and $X_i \; R \; X_{i+1}$ for all $i < k$, we have $\theta(X_k) = \textit{true}$.

Take some set of variables $X_0, \cdots, X_n$ such that $X_0 = X$ and $X_i \; R \; X_{i+1}$ for all $i < n$. Then, by the induction hypothesis, $\theta(X_k) = \textit{true}$. Thus, since $X_k \; R \; X_n$, and $\theta \in \Theta_R$, we can conclude that also $\theta(X_n) = \textit{true}$.

$\leftarrow$ Assume that $\theta \in \Theta_{R^+}$. Furthermore, take some variable X such that $\theta(X) = \textit{true}$. Now take any variable Y such that $(X, Y) \in R$. Since $(X, Y) \in R$, also $(X, Y) \in R^+$. Therefore, since $\theta \in \Theta_{R^+}$, also $\theta(Y) = \textit{true}$. Thus $\theta \in \Theta_R$. $\square$

Given a propositional variable X and propositional relation R, we wish to define an environment which assign *true* to X and is also consistent with R. Not all environments satisfy this condition, if there exists a variable Y such that $(X, Y) \in R$, then the environment assigning *false* to all variables except for X is not consistent with R. On the other hand, the empty environment is consistent with R, but it does not assign *true* to X. However, using the notion of reachability we can define an environment which is consistent with some relation R, and (at least) assigns *true* to some propositional variable X.

Definition 2.2.10. Given a relation R on propositional variables, and propositional variable X. The *minimal environment* θ *under* R such that X is assigned *true*, written $\theta_{R,X}$, is the environment assigning *true* to X and all variables reachable from X through R, and *false* to all other variables.

This definition is such that for any relation R and propositional variable X, $\theta_{R,X}$ is consistent with R.

Lemma 2.2.11. For any relation R on propositional variables and propositional variable X, $\theta_{R,X}$ is consistent with R.

Proof. Let R be a relation on propositional variables, and let X, Y, Z be propositional variables such that $(Y, Z) \in R$ and $\theta_{R,X}(Y) = \mathit{true}$. We need to show that $\theta_{R,X}(Z) = \mathit{true}$.

If $Y = Z$ then this is obviously the case. If $Y = X$ then Z is reachable from X and thus $\theta_{R,X}(Z) = \mathit{true}$.

Now assume that $Y \neq Z$ and $Y \neq X$. In this case, from the definition of minimal environment for variables, Y is reachable from X through R. Thus, there exist variables $X_0, \cdots, X_n$ for some $n > 0$ such that $X_0 = X$, $X_n = Y$ and for all $i < n$ we have $X_i \; R \; X_{i+1}$. But then, Z is reachable from X through R, since we have $X_0, \cdots, X_{n+1}$, with $X_0 = X$ and $X_{n+1} = Z$ such that for all $i < n + 1$ we have $X_i \; R \; X_{i+1}$. Therefore, $\theta_{R,X}(Z) = \mathit{true}$. $\square$

The minimal environment for some variable X under some relation R gives us a simple tool for determining relative consequences between X and other variables thanks to the following property.

Lemma 2.2.12. For any relation R on propositional variables, variable X and propositional formula f, $X \overset{R}{\Rightarrow} f$ iff $[\![f]\!]\theta_{R,X} = \mathit{true}$.

Proof. Take some relation R on propositional variables, variable X and propositional formula f.

- $\rightarrow$ Assume that $X \overset{R}{\Rightarrow} f$. We need to show that $[\![f]\!]\theta_{R,X} = \mathit{true}$. From the definition of the minimal environment for a variable under R, $\theta_{R,X}(X) = \mathit{true}$. Furthermore, from Lemma 2.2.11 $\theta_{R,X}$ is consistent with R, and since $X \overset{R}{\Rightarrow} f$, we have $[\![f]\!]\theta_{R,X} = \mathit{true}$.

- $\leftarrow$ Assume that $[\![f]\!]\theta_{R,X} = \mathit{true}$. We need to show that $X \overset{R}{\Rightarrow} f$. We proceed by induction on the structure of f.

  - $\top$ In this case, obviously for any environment θ consistent with R such that $\theta(X) = \mathit{true}$ we have $[\![\top]\!]\theta = \mathit{true}$, and thus $X \overset{R}{\Rightarrow} f$.

  - $\bot$ In this case, $[\![f]\!]\theta_{R,X} = \mathit{false}$ which contradicts our assumption that $[\![f]\!]\theta_{R,X} = \mathit{true}$.

Y In this case, $[\![Y]\!]\theta_{R,X} = \mathit{true}$, and thus either Y is equal to X, or (from the definition of $\theta_{R,X}$) Y is reachable from X through R. If Y is equal to X then obviously $X \overset{R}{\Rightarrow} Y$.

If Y is reachable from X through R then there exist variables $X_0, \cdots, X_n$, with $X_0 = X$ and $X_n = Y$, such that for all $i < n$ we have $X_i\ R\ X_{i+1}$. We will prove that, for any $n > 0$ and any set of variables variables $X_0, \cdots, X_n$ such that $X_0 = X$ and $X_i\ R\ X_{i+1}$ for all $i < n$, we have $X \overset{R}{\Rightarrow} X_n$. We prove this by induction on n.

$n = 1$ In this case, $X\ R\ X_n$ and thus for any environment θ consistent with R, we have that $\theta(X) = \mathit{true}$ implies $\theta(X_n) = \mathit{true}$, and thus $X \overset{R}{\Rightarrow} X_n$.

$n = k + 1$ We have the following induction hypothesis: For any $k + 1$ variables $X_0, \cdots, X_k$ such that $X_0 = X$ and $X_i\ R\ X_{i+1}$ for all $i < k$, we have $X \overset{R}{\Rightarrow} X_k$.

Given some set of variables $X_0, \cdots, X_n$ such that $X_0 = X$ and $X_i\ R\ X_{i+1}$ for $i < n$, we need to show that $X_0 \overset{R}{\Rightarrow} X_n$.

Take any environment θ consistent with R such that $\theta(X_0) = \mathit{true}$. We need to show that $\theta(X_n) = \mathit{true}$. By the induction hypothesis $X_0 \overset{R}{\Rightarrow} X_k$, and thus $\theta(X_k) = \mathit{true}$. But then, since $X_k\ R\ X_n$, also $\theta(X_n) = \mathit{true}$. Thus we can conclude $X_0 \overset{R}{\Rightarrow} X_n$.

$f_1 \wedge f_2$ In this case, the induction hypothesis gives us that if $[\![f_1]\!]\theta_{R,X} = \mathit{true}$ then $X \overset{R}{\Rightarrow} f_1$, and similarly for f_2.

Since $[\![f]\!]\theta_{R,X} = \mathit{true}$, both $[\![f_1]\!]\theta_{R,X} = \mathit{true}$ and $[\![f_2]\!]\theta_{R,X} = \mathit{true}$. Thus, from the induction hypothesis we have $X \overset{R}{\Rightarrow} f_1$ and $X \overset{R}{\Rightarrow} f_2$.

Take any environment θ consistent with R such that $\theta(X) = \mathit{true}$. Then also $[\![f_1]\!]\theta = \mathit{true}$ and $[\![f_2]\!]\theta = \mathit{true}$, and thus $[\![f]\!]\theta = \mathit{true}$. Therefore $X \overset{R}{\Rightarrow} f$.

$f_1 \vee f_2$ In this case, the induction hypothesis tells us that if $[\![f_1]\!]\theta_{R,X} = \mathit{true}$ then $X \overset{R}{\Rightarrow} f_1$, and similarly for f_2.

Since $[\![f]\!]\theta_{R,X} = \mathit{true}$, either $[\![f_1]\!]\theta_{R,X} = \mathit{true}$ or $[\![f_2]\!]\theta_{R,X} = \mathit{true}$. Thus, from the induction hypothesis we have either $X \overset{R}{\Rightarrow} f_1$ or $X \overset{R}{\Rightarrow} f_2$. The cases are symmetrical, so we will only discuss $X \overset{R}{\Rightarrow} f_1$.

Take any environment θ consistent with R such that $\theta(X) = \mathit{true}$. Then also $[\![f_1]\!] = \mathit{true}$ and thus $[\![f_1 \vee f_2]\!]\theta = \mathit{true}$. Therefore $X \overset{R}{\Rightarrow} f$. □

A natural question is whether we can lift this notion of minimal environments for propositional variables to minimal environments for propositional formulas, and if such an environment has the same or similar properties. The minimal environment for a propositional formula under some relation is obtained by simply taking the union of the minimal environments for all propositional variables occurring in the formula.

Definition 2.2.13. Given a relation R on propositional variables, and propositional formula f. The minimal environment θ under R for f is the union of the minimal environments of all variables from $\mathcal{V}(f)$ under R.

Observe that, if no variables occur in f, then $\theta_{R,f} = \theta_\mu$.

Obviously, for any formula f and relation R, if f is not equivalent to $\bot$ then $\theta_{R,f}$ assigns *true* to f.

Lemma 2.2.14. For any propositional formula f not equivalent to $\bot$ and relation R on propositional variables, $\theta_{R,f}$ is consistent with R.

Proof. Take some propositional formula f and relation R such that f is not equivalent to $\bot$. If f is equivalent to $\top$, then for all environments θ we have that $[\![f]\!]\theta = \mathit{true}$.

Otherwise, $\theta_{R,f}$ assigns *true* to all variables $X \in \mathcal{V}$. Therefore $\theta_{R,f}$ must also be *true*, since there is no negation in the syntax of propositional formulas. $\square$

Furthermore, since the union of environments consistent with a relation R is also consistent with R, for any propositional formula f we have that $\theta_{R,f}$ is consistent with R.

Lemma 2.2.15. For any relation R on propositional variables and propositional formula f, $\theta_{R,f}$ is consistent with R.

Proof. If f does not contain any variables, then $\theta_{R,f} = \theta_\mu$, and thus from Lemma 2.2.2, $\theta_{R,f}$ is consistent with R. Otherwise, $\theta_{R,f}$ is equivalent to the union of the minimal environments for all variables occurring in f under X, and thus from Lemma 2.2.3, $\theta_{R,f}$ is consistent with R. $\square$

The minimal environment on propositional variables gives us a simple way for determining relative consequences; a variable is assigned *true* by the minimal environment of another variable, iff it is a relative consequence. We would like to also have this same property for the definition of the minimal environment of propositional formulas. However, in general this is not the case. Consider the following example.

Example 2.2.16. Take propositional variables X, Y and let R be the empty relation. Let $f = X \vee Y$ and $g = Y$. Then $[\![g]\!]\theta_{R,f} = \mathit{true}$. However, consider $\theta_{R,X}$. This environment is consistent with R. Furthermore, since $\theta_{R,X}(X) = \mathit{true}$, we have $[\![f]\!]\theta_{R,X} = \mathit{true}$. However, $[\![g]\!]\theta_{R,X} = \mathit{false}$, and thus we do not have $f \overset{R}{\Rightarrow} g$.

For clauses not equivalent to $\bot$, we do have this property. Given a clause f not equivalent to $\bot$, and relation on propositional variables R, a propositional formula evaluates to *true* under the minimal environment for f under R iff it is a consequence of f relative to R.

Lemma 2.2.17. For any relation R on propositional variables, clause f not equivalent to $\bot$ and propositional formula g, $[\![g]\!]\theta_{R,f} = \mathit{true}$ iff $f \overset{R}{\Rightarrow} g$.

Proof. Take some relation R on propositional variables, clause f not equivalent to $\bot$ and propositional formula g.

$\rightarrow$ Assume that $[\![g]\!]\theta_{R,f} = \mathit{true}$. We need to show that $f \overset{R}{\Rightarrow} g$. We proceed by induction on the structure of g.

$\top$ In this case trivially $f \overset{R}{\Rightarrow} g$.

$\perp$ This contradicts with our assumption that $[\![g]\!]\theta_{R,f} = \mathit{true}$.

Y In this case, $[\![g]\!]\theta_{R,f} = \theta_{R,f}(Y) = \mathit{true}$. Thus, either $Y \in \mathcal{V}(f)$, or there is some variable $X \in \mathcal{V}(f)$ such that Y is reachable from X through R. In the first case $f \Rightarrow Y$, and thus $f \overset{R}{\Rightarrow} Y$. In the second case, $X \overset{R}{\Rightarrow} Y$, and therefore since $f \overset{R}{\Rightarrow} X$ also $f \overset{R}{\Rightarrow} Y$.

$g_1 \wedge g_2$ In this case we have the following induction hypotheses; if $[\![g_1]\!]\theta_{R,f} = \mathit{true}$ then $f \overset{R}{\Rightarrow} g_1$, and similarly for g_2.

Furthermore, both $[\![g_1]\!]\theta_{R,f} = \mathit{true}$ and $[\![g_2]\!]\theta_{R,f} = \mathit{true}$. Take any environment θ consistent with R such that $[\![f]\!]\theta = \mathit{true}$. We need to show that $[\![g_1 \wedge g_2]\!]\theta = \mathit{true}$. This is the case if both $[\![g_1]\!]\theta = \mathit{true}$ and $[\![g_2]\!]\theta = \mathit{true}$, which follows from the induction hypotheses.

$g_1 \vee g_2$ In this case we have the following induction hypotheses; if $[\![g_1]\!]\theta_{R,f} = \mathit{true}$ then $f \overset{R}{\Rightarrow} g_1$, and similarly for g_2.

Furthermore, either $[\![g_1]\!]\theta_{R,f} = \mathit{true}$ or $[\![g_2]\!]\theta_{R,f} = \mathit{true}$. The cases are symmetrical: Take any environment θ consistent with R such that $[\![f]\!]\theta = \mathit{true}$. We need to show that $[\![g_1 \vee g_2]\!]\theta = \mathit{true}$. This is the case if $[\![g_1]\!]\theta = \mathit{true}$ resp. $[\![g_2]\!]\theta = \mathit{true}$, which follows from the induction hypotheses.

$\leftarrow$ Assume that $f \overset{R}{\Rightarrow} g$. We need to show that $[\![g]\!]\theta_{R,f} = \mathit{true}$.

From Lemma 2.2.15, $\theta_{R,f}$ is consistent with R. Furthermore, from the definition of $\theta_{R,f}$, since f is not equivalent to $\perp$, $[\![f]\!]\theta_{R,f} = \mathit{true}$. Therefore, $[\![g]\!]\theta_{R,f} = \mathit{true}$. □

We used the notion of consistency on environments to lift relations on propositional variables to relations on propositional formulas. For this we introduced the notion of relative consequence. One could wonder how this notion of relative consequence behaves on propositional variables compared to propositional formulas. A nice property to have, would be a one to one correspondence between relative consequence on propositional variables, and membership in a relation; i.e. given some relation R on propositional variables, can we derive $X \overset{R}{\Rightarrow} Y$ from $(X, Y) \in R$ and/or vice versa? One direction is obvious; given a relation on propositional variables R and variables $(X, Y) \in R$, we then also have $X \overset{R}{\Rightarrow} Y$. However, the converse stating that if $X \overset{R}{\Rightarrow} Y$ then we can conclude $(X, Y) \in R$ does not hold. Consider the following example.

Example 2.2.18. Let $R = \{(X, Z), (Z, Y)\}$, then $X \overset{R}{\Rightarrow} Y$ (for any environment consistent with R, if X is a assigned *true* then so is Z, and if Z is assigned *true* then so is Y). However, we do not have $(X, Y) \in R$.

Despite the fact that we cannot derive membership in a relation from relative consequence between variables, we do have a weaker property. For propositional variables X, Y and relation on propositional variables R, we have that $X \overset{R}{\Rightarrow} Y$ iff X, Y is in the reflexive, transitive closure of R.

Lemma 2.2.19. For all propositional variables X, Y and relations on propositional variables R, $X \overset{R}{\Rightarrow} Y$ if and only if $X \; R^* \; Y$.

Proof. Take some relation R on propositional variables.

- $\rightarrow$ Take some X, Y such that $X \overset{R}{\Rightarrow} Y$. Then, from Lemma 2.2.12 we have that $\theta_{R,X}(Y) = \mathit{true}$. But then, either $X = Y$ or Y is reachable from X through R. In the first case trivially $(X, Y) \in R^*$. In the second case, $(X, Y) \in R^+$, and therefore also $(X, Y) \in R^*$.
- $\leftarrow$ Take some $(X, Y) \in R^*$. If $X = Y$ then obviously $X \overset{R}{\Rightarrow} Y$. Assume that $X \neq Y$. We need to show that $X \overset{R}{\Rightarrow} Y$. In this case, $(X, Y) \in R^+$ and thus according to Lemma 2.2.8 we know that Y is reachable from X through R. But then by definition of $\theta_{R,X}$, $\theta_{R,X}(Y) = \mathit{true}$, and thus from Lemma 2.2.12 we know $X \overset{R}{\Rightarrow} Y$. □

2.3 BES

Boolean equation systems, or BES for short, are a formalism for encoding model checking problems and other decision problems [18]. BES are usually defined as lists of fixed point equations, though we will use a slightly different definition in terms of lists of blocks, where a block is a lists of Boolean equations and a fixpoint symbol.

Definition 2.3.1. A *Boolean equation* is a pair of a propositional variable X and propositional formula f, written as $(X = f)$.

Given a list of Boolean equations B, we will write $\mathbf{bnd}_B$ to indicate the set of all propositional variables occurring on the left-hand side of equations in B. If for each variable $X \in \mathbf{bnd}_B$ there exists only one Boolean equation in B with X at the left hand side, then we say that B is *well-formed.* Furthermore, for well-formed B and any variable $X \in \mathbf{bnd}_B$, we will write f_X to indicate the right-hand side of the equation $(X = f)$ in B. Finally, if no confusion is possible then we may write $\mathbf{bnd}$ instead of $\mathbf{bnd}_B$.

As mentioned, we will define BES as lists of blocks. A block is a non-empty lists of Boolean equations paired with a fixpoint symbol.

Definition 2.3.2. A block $\mathcal{B}$ is defined as a pair of a non-empty list of Boolean equations B and a fixpoint symbol (either μ or ν).

The following is an example of a block.

Example 2.3.3.

$$(\mu\langle(X = A \wedge B)(Y = X \vee (Y \wedge \top))\rangle)$$

We will typically write σB to indicate a block, where $\sigma \in \{\mu, \nu\}$.

A *Boolean Equation System* (or *BES*) is a list of blocks with alternating fixpoint symbols.

Definition 2.3.4. We recursively define a BES as

$$\mathcal{E} ::= \mathcal{E}_\nu \mid \mathcal{E}_\mu$$

Where $\mathcal{E}_\nu$ and $\mathcal{E}_\mu$ are defined as

$$\mathcal{E}_\mu ::= \varepsilon \mid (\mu B)\mathcal{E}_\nu$$
$$\mathcal{E}_\nu ::= \varepsilon \mid (\nu B)\mathcal{E}_\mu$$

With ε being the empty list, and B being a non-empty list of Boolean equations.

In this text, we will typically use $\mathcal{E}$ to indicate a BES. For an arbitrary equation system $\mathcal{E}$, we lift the definition of **bnd**, f_X and well-formed in the natural way from lists of Boolean equations to blocks and lists of blocks. From now on, we will only consider well-formed BES in this text.

Given a BES, variables can be grouped together based on their rank, which is a measure of the nesting depth of the variable.

Definition 2.3.5. Let $\mathcal{E}$ be a BES and let X be some variable. The *rank* of X in $\mathcal{E}$, written $\mathbf{rank}_{\mathcal{E}}(X)$ is 0 if X is not in $\mathbf{bnd}_{\mathcal{E}}$. Otherwise, it is equal to the number of blocks up to and including the block in which X is bound, counting from 1 if the first block in $\mathcal{E}$ has a least fixpoint symbol and 2 otherwise.

Given a variable X and BES $\mathcal{E}$, if no confusion is possible then we will write $\mathbf{rank}(X)$ instead of $\mathbf{rank}_{\mathcal{E}}(X)$. The following is an example of the ranks of variables in a BES.

Example 2.3.6. Consider the following BES:

$$(\nu\langle(X = f_X)(Y = f_Y)\rangle)(\mu\langle(Z = f_Z)\rangle)$$

Then the ranks of A, X, Y, Z are as follows:

$$\mathbf{rank}(A) = 0 \quad \mathbf{rank}(X) = 2 \quad \mathbf{rank}(Y) = 2 \quad \mathbf{rank}(Z) = 3$$

2.3.1 The semantics of BES

We start this section with some fixpoint theory. A *poset* $(A, \leq)$, is a set A paired with a binary relation on the set $\leq \subseteq A \times A$, such that $\leq$ is reflexive, antisymmetric and transitive. A poset $(A, \leq)$ is a complete lattice if all subsets of A have both a *supremum* (least upper bound) and an *infimum* (greatest lower bound).

Take some arbitrary complete lattice $(A, \leq)$ with least and greatest elements A_ν and A_μ, $\sqcup$ for determining the supremum and $\sqcap$ for determining the infimum. Furthermore, let $f : A \to A$ be a function on A. A fixed point of f is an element $a \in A$ such that $f(a) = a$. The least fixed point of f, indicated with μf, is the fixed point of f such that, for all other fixed points a of f, $\mu f \leq a$. The greatest fixed point of f, written νf, is the fixed point of f such that, for all other fixed points a of f, $a \leq \nu f$. Furthermore, we say that f is *monotone* iff, for all elements $a, b \in A$, if $a \leq b$ then also $f(a) \leq f(b)$. According to Tarski's Theorem [22], if f is monotone then least and greatest fixed points of f are guaranteed to exist, namely $\mu f = \sqcap\{a \in A \mid f(a) \leq a\}$ resp. $\nu f = \sqcup\{a \in A \mid f(a) \geq a\}$. Furthermore, the least and greatest fixed points of f can be obtained using a transfinite approximation; to this end, we define the *approximant terms* of f (see also [21]).

Definition 2.3.7. For ordinal α and limit ordinal λ, the approximant $\sigma^\alpha f$ is defined by transfinite induction as follows.

$$\begin{aligned}
\sigma^0 f &= A_\sigma \\
\sigma^{\alpha+1} &= f(\sigma^\alpha f) \\
\sigma^\lambda f &= \begin{cases} f(\sqcup_{\alpha<\lambda}\sigma^\alpha f) & \text{if } \sigma = \mu \\ f(\sqcap_{\alpha<\lambda}\sigma^\alpha f) & \text{if } \sigma = \nu \end{cases}
\end{aligned}$$

The following is a method for determining the least and greatest fixpoint of a monotone function.

Lemma 2.3.8. Let *Ord* be the class of all ordinals. Then we can obtain the least and greatest fixpoints of a monotone function as follows.

$$\mu f = \sqcup_{\alpha \in Ord}(\mu^\alpha f) \quad \nu f = \sqcap_{\alpha \in Ord}(\nu^\alpha f)$$

We are interested in the following posets. Take some natural number n, and consider $(\mathbb{B}^n, \sqsubseteq^n)$, with $\sqsubseteq^n\colon \mathbb{B}^n \to \mathbb{B}^n$ such that for any $\bar{b}_1, \bar{b}_2 \in \mathbb{B}^n$, $\bar{b}_1 \sqsubseteq^n \bar{b}_2$ if and only if, for all $1 \leq i \leq n$, $(\bar{b}_1)_i$ (the i-the element of $\bar{b}_1$) implies $(\bar{b}_2)_i$. Then $(\mathbb{B}^n, \sqsubseteq^n)$ is a complete lattice, with minimal and maximal elements the n-tuples with all elements *false* resp. *true* (which we will call $\bar{b}_\mu$ resp. $\bar{b}_\nu$). Furthermore, for any monotone function $f : \mathbb{B}^n \to \mathbb{B}^n$, since elements from tuples from $\mathbb{B}^n$ can only take one of two values, and there are always n elements in tuples from $\mathbb{B}^n$, a fixed point of f is always reached within n iterations of f, and thus $\sigma f = f^n(\bar{b}_\sigma)$, where $f^n(\bar{b})$ is defined as follows.

Definition 2.3.9. Given a function $f : A \to A$, repeatedly applying f n times to a and n times to some element $a \in A$ is defined as follows.

$$\begin{aligned}
f^0(a) &= a \\
f^{k+1}(a) &= f(f^k(a))
\end{aligned}$$

The syntax of BES is usually defined as a list of fixpoint equations, where a fixpoint equation is a Boolean equation paired with a least or greatest fixpoint symbol, see also [18].

In this text however, we have defined the syntax of BES as lists of blocks. Rephrasing the semantics of BES defined using lists of fixpoint equations, to BES defined using lists of blocks (with alternating signs) is standard. It relies on Bekič theorem [2] for converting nested fixed points to simultaneous fixed points.

Definition 2.3.10. For any BES $\mathcal{E}$ and environment θ, we recursively define the semantics of $\mathcal{E}$, written $(\!|\mathcal{E}|\!)\theta$, as follows.

$$\begin{aligned}
(\!|\varepsilon|\!)\theta &= \theta \\
(\!|(\sigma B)\mathcal{E}'|\!)\theta &= (\!|\mathcal{E}'|\!)(\theta[\langle X_i\rangle_{i=1}^n := \sigma(\mathcal{F}(\mathcal{E}', B, \theta))])
\end{aligned}$$

Where $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$ and $\mathcal{F}$ is defined as follows.

$$\mathcal{F}(\mathcal{E}', B, \theta) := \lambda\langle b_i\rangle_{i=1}^n \in \mathbb{B}^n.\langle [\![f_{X_i}]\!]((\!|\mathcal{E}'|\!)(\theta[\langle X_j := b_j\rangle_{j=1}^n]))\rangle_{i=1}^n$$

Now, for any BES $\mathcal{E} = (\sigma B)\mathcal{E}'$, with B a list of n Boolean equations, $\mathcal{F}(\mathcal{E}', B, \theta)$ is monotone in $(\mathbb{B}^n, \sqsubseteq^n)$, and therefore least and greatest fixed points are guaranteed to exist. In fact, $\sigma(\mathcal{F}(\mathcal{E}', B, \theta)) = (\mathcal{F}(\mathcal{E}', B, \theta))^n(b_\sigma)$.

In Coq, the type of $\mathcal{F}$ would be $\mathbb{B}^n \to \mathbb{B}^n$. However, the type of $\mathbb{B}^n$ is dependent on parameter n (see also [1] for more on dependent types), and dependent types are somewhat more complex to work with. Instead, we will present a semantics for BES which uses a function $F : (\mathcal{X} \to \mathbb{B}) \to (\mathcal{X} \to \mathbb{B})$ on environments instead of tuples of Booleans, which we have obtained via a simple conversion from n tuples to environments, and we will show that this semantics is equivalent to the standard semantics.

Before we introduce this alternative semantics of BES we first introduce a short hand for unfolding a block within an environment.

Definition 2.3.11. Let $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$ be a list of n of Boolean equations. Given an environment θ, we define the unfolding of B in θ, written $||B||\theta$, as follows.

$$||B||\theta = \theta[(X_1, \cdots, X_n) := ([\![f_{X_1}]\!]\theta, \cdots, [\![f_{X_n}]\!]\theta)]$$

Here, $(X_1, \cdots, X_n) := ([\![f_{X_1}]\!]\theta, \cdots, [\![f_{X_n}]\!]\theta)$ *denotes the simultaneous redefining of* θ *in* $X_1, \cdots, X_n$ *to the values* $[\![f_{X_1}]\!]\theta, \cdots, [\![f_{X_n}]\!]\theta$*.*

Given an environment θ and a block $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$ with for all $0 \leq i, j \leq n$ such that $i \neq j$ we have $X_i \neq X_j$. Then we can obtain an n-tuple of Booleans $\bar{b}$, by defining the value of each of its indexes i: $\bar{b}_i = \theta(X_i)$. Conversely, given an n-tuple of Booleans $\bar{b}$. We can 'store' this tuple in an environment θ, by defining some block $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$ such that, for all $0 \leq i, j \leq n$ with $i \neq j$ we have $X_i \neq X_j$. We can then store $\bar{b}$ in θ by taking $\theta[\langle X_i = \bar{b}_i\rangle_{i=1}^n]$.

Using this method for converting from tuples of Booleans to environments, and the definition of unfolding a block within an environment, the semantics of BES can also be formulated as follows.

Definition 2.3.12. We define the semantics of a BES under an environment θ, written $[\![\mathcal{E}]\!]\theta$, as follows.

$$[\![\varepsilon]\!]\theta = \theta$$
$$[\![(\sigma B)\mathcal{E}']\!]\theta = [\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n])$$

Where $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$ and F is defined as follows.

$$F(\mathcal{E}', B, \theta) = \lambda\theta_b.||B||([\![\mathcal{E}']\!](\theta[\langle X_i := \theta_b(X_i)\rangle_{i=1}^n)])$$

The two semantics presented in this part are equivalent, as shown by the following lemma. For the proof of this lemma, we refer the reader to appendix B.

Lemma 2.3.13. For any BES $\mathcal{E}$ and for any environment θ, $(\!|\mathcal{E}|\!)\theta = [\![\mathcal{E}]\!]\theta$.

Chapter 3

Solutions of consequence

Solving a BES involves a complex fixpoint approximation with double recursion. Obviously, the larger the BES, the more expensive solving the BES becomes. To somewhat mitigate this exponential blowup, methods of abstraction have been defined, which are used to reduce the complexity of BES. However, the soundness proofs of these abstraction methods are laborious and complex. Thus the consistent correlation has been created to simplify such soundness proofs [23]. This relation was defined such that, by showing that the abstraction technique preserved a consistent correlation between the abstract and the concrete BES, the soundness of the abstraction technique would immediately follow.

At the basis of the consistent correlation lies the consistent consequence relation. Furthermore, the consistent consequence relation is in itself a method of abstraction on BES. It allows one to relate the solution of variables from (possibly different) BES, without solving the BES themselves. Given a BES $\mathcal{E}$, if we can relate a pair of bound variables via a consistent consequence relation on $\mathcal{E}$, then we know that if the solution of a BES is *true* for one variable, then it must also be *true* for the other variable, i.e. in the solution of $\mathcal{E}$ one variable is a logical consequence of the other variable, and therefore it would be sufficient to solve the BES for one variable to obtain some knowledge about the solution of the BES for the other variable.

In this chapter, we introduce the consistent consequence relation, as well as the main theorem capturing the relation between the consistent consequence relation and the solution of a BES.

3.1 The consistent consequence relation

Given a BES, a relation on propositional variables is a consistent consequence relation on this BES if, for all pairs of variables related, the rank of the variables is equal and, if both variables are bound in the BES, then the right hand sides of the variables should be consequences relative to the relation.

Definition 3.1.1. Let $\mathcal{E}$ be a BES, and R a relation on propositional variables. R is a *consistent consequence on* $\mathcal{E}$ if, for all variables X, Y such that $X \; R \; Y$, we have:

1. $\mathbf{rank}(X) = \mathbf{rank}(Y)$

2. if $X, Y \in \mathbf{bnd}$ then $f_X \overset{R}{\Rightarrow} f_Y$

Given an equation system $\mathcal{E}$, for any pair of propositional variables X, Y, if there exists a consistent consequence relation R relating only variables bound in $\mathcal{E}$ (i.e. $R \subseteq \mathbf{bnd}_{\mathcal{E}} \times \mathbf{bnd}_{\mathcal{E}}$) such that $X\ R\ Y$, then we say that Y is a consistent consequence of X, written as $X \lessdot^{\mathcal{E}} Y$.

Given two consistent consequences, their union is also a consistent consequence.

Lemma 3.1.2. [10] Given a BES $\mathcal{E}$ and consistent consequence relations R_1, R_2 on this BES. Then, the union of R_1 and R_2, written as $R_1 \cup R_2$, is also a consistent consequence relation.

From the previous lemma, we can derive that there exists some largest consistent consequence relation (if we restrict the relation to only relate bound variables). In fact, $\lessdot^{\mathcal{E}}$ is the largest consistent consequence relation on the bound variables of $\mathcal{E}$.

Lemma 3.1.3. [10] Given a BES $\mathcal{E}$, $\lessdot^{\mathcal{E}}$ is the largest consistent consequence relation relating only variables in $\mathcal{E}$.

If no confusion is possible we may write $\lessdot$ instead of $\lessdot^{\mathcal{E}}$.

Lemma 3.1.4. [10] Given a BES $\mathcal{E}$, $\lessdot^{\mathcal{E}}$ is a preorder.

3.1.1 Some examples

On any BES, the simplest consistent consequence relation is the empty relation. For any BES, the empty relation trivially satisfies the properties of a consistent consequence relation. The following are a few less trivial examples of BES and their (largest) consistent consequence relations.

Example 3.1.5. Consider the following BES.

$$(\mu\langle(X_0 = \bot)(X_1 = X_1)\rangle)$$

Then $\lessdot\ = \{(X_0, X_0), (X_0, X_1), (X_1, X_1)\}$. Observe that we do not have $X_1 \lessdot X_0$. Consider θ_ν, which assigns *true* to all variables. This environment is consistent with R, however, $[\![f_{X_1}]\!]\theta = [\![X_1]\!]\theta = \mathit{true}$ and $[\![f_{X_0}]\!]\theta = [\![\bot]\!]\theta = \mathit{false}$, and thus not $f_{X_1} \overset{R}{\Rightarrow} f_{X_0}$.

Example 3.1.6. Consider the following BES.

$$\begin{aligned}(\mu\langle(X_0 = X_4)(X_1 = X_0 \vee X_2)(X_2 = X_4 \vee X_5)(X_3 = X_2)(X_4 = X_4)\rangle)\\ (\nu\langle(X_5 = X_5)\rangle)\end{aligned}$$

Then we have (among others) $X_0 \lessdot X_2$, $X_1 \lessdot X_3$, $X_3 \lessdot X_1$ and $X_4 \lessdot X_0$. Showing this is somewhat complicated as it requires defining consistent consequence relations relating each of the pairs of variables. However, in the next chapter, we will introduce a sound and complete proof system for deriving consistent consequences, and we will show that it is capable of deriving each of these.

3.2 An important theorem

In the introduction to this section, we stated that there is a relation between consistent consequences on a BES and its solution. The following claim gives this relation. It states that, given a BES and a pair of variables (X, Y), if X is related to Y by a consistent consequence relation on the BES, then the solution of X in the BES implies the solution of Y in the BES.

Claim [5],[10] Let $\mathcal{E}$ be a BES and R a consistent consequence relation on $\mathcal{E}$. Then, for all environments θ consistent with R, $[\![\mathcal{E}]\!]\theta$ is also consistent with R.

This claim shows us why we are interested in consistent consequence relations. We can use consistent consequence relations to show a dependency between the solution of one variable in a BES and the solution of another variable in this BES, without solving either BES. In particular, this holds for closed BES, as the following Corollary states.

Corollary 3.2.1. Given a closed BES $\mathcal{E}$. For all variables X, Y such that $X \lessdot Y$ and any environment θ, $([\![\mathcal{E}]\!]\theta)(X) = \mathit{true}$ implies $([\![\mathcal{E}]\!]\theta)(Y) = \mathit{true}$.

The usefulness of this property is best demonstrated via an example.

Example 3.2.2. Consider the following BES:

$$\mathcal{E} = (\nu\langle(X = X)(Y_0 = Y_1)\cdots(Y_n = Y_0)\rangle)$$

Then we can easily solve X in $\mathcal{E}$ for any θ; for any θ, we have $([\![\mathcal{E}]\!]\theta)(X) = \mathit{true}$, since the solution of X only depends on X, and not on any of the other equations in the BES. However, the solution of Y_0 depends on all other Y_i, and thus solving $\mathcal{E}$ for Y_0 is a more complex procedure.

However, $X \lessdot Y_0$. Thus, if the previous claim is correct, then for any $\theta \in \Theta_{\lessdot}$, we have $([\![\mathcal{E}]\!]\theta)(X) = \mathit{true}$ implies $([\![\mathcal{E}]\!]\theta)(Y_0) = \mathit{true}$. Furthermore, since we are dealing with a closed BES, we have the following property: For any environments θ, θ', for any BES $\mathcal{E}$, and for any variable X bound in E, $([\![\mathcal{E}]\!]\theta)(X) = ([\![\mathcal{E}]\!]\theta')(X)$. [18] Therefore we can conclude that, for any environment θ, $([\![\mathcal{E}]\!]\theta)(Y_0) = \mathit{true}$. Thus we have obtained the solution of Y_0 without solving $\mathcal{E}$ for Y_0.

To prove the claim made previously, we will show that being consistent with a consistent consequence relation is invariant under each of the operations involved with solving a BES, from which the theorem (Theorem 3.2.6) will follow naturally.

First of, given a BES consisting of a single block (σB), a consistent consequence relation R on this BES and an environment θ consistent with R. Then unfolding B in θ is also consistent with R.

Lemma 3.2.3. Let $\mathcal{E} = (\sigma B)$ be a BES consisting of a single block, and let R be a consistent consequence on $\mathcal{E}$. Then,

$$\forall\theta \in \Theta_R : (||B||\theta) \in \Theta_R$$

Proof. Let $\mathcal{E} = \langle\sigma B\rangle$ be a BES consisting of a single block, and let R be a consistent consequence on $\mathcal{E}$. Furthermore, take some environment $\theta \in \Theta_R$, and propositional variables $(X, Y) \in R$ such that $(||B||\theta)(X) = \mathit{true}$. We need to show that $(||B||\theta)(Y) = \mathit{true}$.

Since R is a consistent consequence relation, and thus only relates variables of the same rank, either both X and Y are bound in B, or neither of them is. The first approximation of a block only modifies variables bound in the block, thus if neither X nor Y is bound in B and $(||B||\theta)(X) = \mathit{true}$, then $\theta(X)$ must also be true. Since $X\ R\ Y$ and $\theta \in \Theta_R$, we then also have $\theta(Y) = \mathit{true}$, and thus $(||B||\theta)(Y) = \mathit{true}$.

If both X and Y are bound in B then $(||B||\theta)(X) = \mathit{true} = [\![f_X]\!]\theta$. Furthermore, since R is a consistent consequence relation, we have $f_X \overset{R}{\Rightarrow} f_Y$. Since θ is consistent with R, we also know that $[\![f_Y]\!]\theta = \mathit{true}$, and therefore, $(||B||\theta)(Y) = [\![f_Y]\!]\theta = \mathit{true}$. □

Next we will show that given a relation R and a function f on environments which maintains consistency with R, repeatedly applying f to some environment θ results in an environment which is still consistent with θ, regardless of the number of applications.

Lemma 3.2.4. Let R be some relation on propositional variables, and let θ be some environment consistent with R. Then, for any function f mapping environments to environments which maintains consistency of environments with R, and for any natural number n, we have that $f^n(\theta)$ is also consistent with R.

Proof. Let R be some relation on propositional variables, let f be a function on environments which maintains consistency of environments with respect to R and let n be some natural number. We need to show that for any environment θ consistent with R, $f^n(\theta)$ is also consistent with R. We will proceed by induction on n.

$n = 0$ Take some θ consistent with R. Then, $f^n(\theta) = f(\theta)$, and since f maintains consistency of environments with R, $f(\theta)$ is consistent with R.

$n = k + 1$ The induction hypothesis is as follows. For any environment θ' consistent with R, we have that $f^k(\theta')$ is also consistent with R.

Take some environment θ consistent with R. Then we have $f^n(\theta) = f(f^k(\theta))$. From the induction hypothesis, we know that $f^k(\theta)$ is consistent with R. Therefore, since f maintains consistency with R, $f^n(\theta)$ is also consistent with R. □

There is one more operation for which we need to show that it maintains consistency, namely redefining an environment for all bound variables from a block with their solution from some other consistent environment.

Lemma 3.2.5. Let $\mathcal{E} = (\sigma B)$ be a BES consisting of a single block such that $B = \langle X_i = f_{X_i}\rangle_{i=1}^{n}$, and let R be a consistent consequence on $\mathcal{E}$. Then for all environments θ_1, θ_2 consistent with R, we have that $\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^{n}]$ is also consistent with R.

Proof. Let $\mathcal{E} = \langle\sigma B\rangle$ be a BES consisting of a single block such that $B = \langle X_i = f_{X_i}\rangle_{i=1}^{n}$, and let R be a consistent consequence on $\mathcal{E}$. Take environments θ_1, θ_2 consistent with R, and propositional variables $(X, Y) \in R$ such that $(\theta_1[\langle X_i :=$

$\theta_2(X_i)\rangle_{i=1}^n])(X) = \mathit{true}$. We need to show that $(\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^n])(Y) = \mathit{true}$.

Since R is a consistent consequence relation on $\mathcal{E}$, X and Y must have the same rank, and therefore either both X and Y are bound in $\mathcal{E}$, or neither of them is.

If neither X nor Y is bound in $\mathcal{E}$, then $(\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^n])(X) = \theta_1(X)$ and $(\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^n])(Y) = \theta_1(Y)$. Since θ_1 is consistent with R, if $\theta_1(X)$ is *true* then so is $\theta_1(Y)$.

If both X and Y are bound in $\mathcal{E}$, then $(\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^n])(X) = \theta_2(X)$ and $(\theta_1[\langle X_i := \theta_2(X_i)\rangle_{i=1}^n])(Y) = \theta_2(Y)$. Since θ_2 is consistent with R, if $\theta_2(X)$ is *true* then so is $\theta_2(Y)$. □

Finally, we can prove the claim which we started this section with. This theorem was formalized in Coq Lemma 6.8.5.

Theorem 3.2.6. Let $\mathcal{E}$ be a BES, and R a consistent consequence relation on $\mathcal{E}$. Then for all $\theta \in \Theta_R$, we have $[\![\mathcal{E}]\!]\theta \in \Theta_R$.

Proof. Let $\mathcal{E}$ be a BES and R a consistent consequence relation on $\mathcal{E}$. We need to show that, for any θ consistent with R, we have that $[\![\mathcal{E}]\!]\theta$ is also consistent with R.

We do this by induction on the number of blocks in $\mathcal{E}$.

$\mathcal{E} = \varepsilon$ In this case, for any θ we have $[\![\mathcal{E}]\!]\theta = \theta$, and thus if an environment θ is consistent with R then so is $[\![\mathcal{E}]\!]\theta$.

$\mathcal{E} = (\sigma B)\mathcal{E}'$ In this case we have the following induction hypothesis: if R is a consistent consequence on $\mathcal{E}'$, then for any environment θ consistent with R we have that $[\![\mathcal{E}']\!]\theta$ is also consistent with R.

First, we prove that R is also a consistent consequence on $\mathcal{E}'$. Take any variables $(X, Y) \in R$. We need to show that $\mathbf{rank}_{\mathcal{E}'}(X) = \mathbf{rank}_{\mathcal{E}'}(Y)$ and, if X and Y are bound in $\mathcal{E}'$, then the right hand side of Y in $\mathcal{E}'$ is a consequence of the right hand side of X in $\mathcal{E}'$, relative to R.

Since $X\ R\ Y$, and R is a consistent consequence on $\mathcal{E}$, the ranks of X and Y in $\mathcal{E}$ are equal. Therefore, either both X and Y are not bound in $(\sigma B)\mathcal{E}'$, or they are both bound in B, or they are both bound in $\mathcal{E}'$. In the first two cases, we have $\mathbf{rank}_{\mathcal{E}'}(X) = 0 = \mathbf{rank}_{\mathcal{E}'}(Y)$. In the later case, we have $\mathbf{rank}_{\mathcal{E}}(X) = \mathbf{rank}_{\mathcal{E}'}(X) + 1$ (if $\sigma = \nu$) and $\mathbf{rank}_{\mathcal{E}}(X) = \mathbf{rank}_{\mathcal{E}'}(X) - 1$ otherwise. Similarly for the rank of Y. Thus, since $\mathbf{rank}_{\mathcal{E}}(X) = \mathbf{rank}_{\mathcal{E}}(Y)$, we have $\mathbf{rank}_{\mathcal{E}'}(X) = \mathbf{rank}_{\mathcal{E}'}(Y)$.

Next, assume that both X and Y are bound in $\mathcal{E}'$. We need to show that the right hand side of Y in $\mathcal{E}'$ is a consequence of the right hand side of Y in $\mathcal{E}'$ relative to R. Observe that, since X is bound in $\mathcal{E}'$, it is not bound in (σB) (since $\mathcal{E}$ is well-formed), and thus the right hand side of X in $\mathcal{E}$ is the same as the right hand side of X in $\mathcal{E}'$. The same holds for Y. Let f_X and f_Y be these right hand sides for X resp. Y. Then, since R is a consistent consequence relation on $\mathcal{E}$ relating X to Y, we have $f_X \overset{R}{\Rightarrow} f_Y$.

Thus we can conclude that R is a consistent consequence relation on $\mathcal{E}'$, and therefore the induction hypothesis tells us that for any environment θ consistent with R, $[\![\mathcal{E}']\!]\theta$ is also consistent with R.

Furthermore R is also a consistent consequence relation on the BES consisting of the single block B. This can be shown using a reasoning similar to proving that R is a consistent consequence relation on $\mathcal{E}'$.

Note that B is some non-empty list of Boolean equations, thus assume that $B = \langle X_i = f_{X_i} \rangle_{i=1}^n$. Next, take some environment θ consistent with R. We need to show that $[\![(\sigma B)\mathcal{E}']\!]\theta$ is also consistent with R.

From the definition of the solution of a BES, we obtain
$[\![(\sigma B)\mathcal{E}']\!]\theta = [\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n])$.

From the induction hypothesis,
$[\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n])$ is consistent with R if
$\theta[\langle X_i := ((F(\mathcal{E}'\ B\ \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n]$ is consistent with R.

From Lemma 3.2.5, since θ is consistent with R,
$\theta[\langle X_i := ((F(\mathcal{E}'\ B\ \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n]$ is consistent with R if
$(F(\mathcal{E}', B, \theta))^n(\theta_\sigma)$ is consistent with R.

Furthermore, from Lemma 2.2.2 we know that θ_σ is consistent with R, and thus from Lemma 3.2.4, $(F(\mathcal{E}', B, \theta))^n(\theta_\sigma)$ is consistent with R if $F(\mathcal{E}', B, \theta)$ maintains consistency.

From the definition of the semantics of a BES, we obtain $F(\mathcal{E}', B, \theta) = \lambda\theta_b.||B||([\![\mathcal{E}']\!](\theta[\langle X_i := \theta_b(X_i)\rangle_{i=1}^n]))$.

Take any environment θ_b consistent with R. We need to show that $||B||([\![\mathcal{E}']\!](\theta[\langle X_i := \theta_b(X_i)\rangle_{i=1}^n]))$ is also consistent with R. From Lemma 3.2.3, this is the case if $[\![\mathcal{E}']\!](\theta[\langle X_i := \theta_b(X_i)\rangle_{i=1}^n])$ is consistent with R.

From the induction hypothesis, since R is a consistent consequence relation on $\mathcal{E}'$, this is the case if $\theta[\langle X_i := \theta_b(X_i)\rangle_{i=1}^n]$ is consistent with R. Finally, this is the case according to Lemma 3.2.5, since both θ and θ_b are consistent with R. $\square$

3.3 On variations of consistent consequences

A few variations have occurred in the literature for defining consistent consequence relations. In particular, the relation has been defined both in a setting of BES and in a more generalized setting of PBES. In [10] the consistent consequence relation was defined on BES as follows.

Definition 3.3.1. Let $\mathcal{E}$ be a BES, and R a relation on propositional variables. R is a *consistent consequence on* $\mathcal{E}$ if, for all variables X, Y such that $X\ R\ Y$ and X and Y are bound in $\mathcal{E}$, we have:

1. $\mathbf{rank}(X) = \mathbf{rank}(Y)$
2. $f_X \overset{R}{\Rightarrow} f_Y$

This definition is similar to the one presented in the first section of this chapter. However the restrictions on the relation are only applied on bound variables (so pairs of unbound variables, and pairs of bound and unbound variables need not have the same rank).

The definition introduced in the first section of this chapter, which we will use throughout this document, is more in line with the definition from [5] (though it is defined on BES instead of PBES). In fact, we found an issue with the definition from [10] (given above). In [10], the claim we made in the previous section, which gives the relation between the notion of consistent consequence and the solution of BES, is proven. However, we found a small issue with this proof. This definition allows unbound variables to relate to bound variables, which causes contradictions with the claim, as demonstrated in the following example.

Example 3.3.2. Consider the following BES.

$$\mathcal{E} = (\mu\langle X = X\rangle)$$

Let $R = \{(A, X)\}$ for some propositional variable A. Then, according to definition 3.3.1, R is a consistent consequence relation on $\mathcal{E}$. Now consider an environment θ with $\theta(X) = \theta(A) = \mathit{true}$. Then θ is consistent with R. However, $[\![\mathcal{E}]\!]\theta$ is not consistent with R, since $([\![\mathcal{E}]\!]\theta)(X) = \mathit{false}$ but $([\![\mathcal{E}]\!]\theta)(A) = \mathit{true}$. Thus, the relation between the semantics of BES, as proven in Theorem 3.2.6, and the notion of consistent consequence does not hold using this definition.

By taking a notion of consistent consequence more in line with [5], we avoid these issues. The difference lies in the fact that we also allow unbound variables to be related to each other (but *not* unbound variables to bound variables or vice versa). This relaxation is required to be able to use the induction hypothesis when proving the relation between the semantics of BES and the notion of consistent consequence, which is the place where the proof in [10] made a mistake.

Chapter 4

A proof system of consequence

The consistent consequence relation presented in the previous chapter has a complex definition in terms of the semantics of a BES. To show that a relation R on propositional variables is a consistent consequence relation for some BES, we need to show two things. The first is that it only relates variables of equal rank, which is not to hard. The second is that the right hand-sides variables related by R are consequences relative to R. For this, one needs to reason on *all* environments consistent with R, which is difficult.

To gain greater insight into the notion of consistent consequence relations, a proof system was created in [10] to syntactically derive that variables in a BES are related via a consistent consequence relation, and it was posed that this proof system was sound and complete for deriving consistent consequences between propositional variables. This proof system was initially created by adding two rules to a proof system for deriving logical consequences on positive propositional formulas, namely rules CC and CNT, the result of which is shown in Table 4.1.

The rules in this table are of the following shape.

$$\frac{\vdash A \subset B}{\Gamma' \vdash C \subset D}\,\text{label}$$

This should be read as 'given that we can conclude B from A in context Γ, then using rule 'label', we can derive that we can conclude D from C in context Γ''. Given a BES $\mathcal{E}$ and propositional variables X, Y, to derive $X \lessdot Y$ using the proof system, one needs to construct a tree such that $\vdash\ X \subset Y$ is in the root, and all leaves are closed.

The proof system shown in Table 4.1 is sufficiently strong to derive consistent consequences for some examples. However, proving that the system is complete in general is more complicated. To simplify the proof of completeness, a third rule was added:

$$\frac{\Gamma \vdash \alpha \subset \beta}{\Gamma \vdash \alpha\eta \subset \beta\eta}\,\text{SUB1}$$

Here, α and β are propositions, η is a substitution mapping propositional variables to propositional formulas, and $f\eta$ is the natural extension of η from variables to terms.

This rule is very similar to a standard rule for deriving logical consequences.

$$\frac{\alpha \subset \beta}{\alpha\eta \subset \beta\eta}\ \text{SUB2}$$

This rule can be derived from the other rules in systems for deriving logical consequence. Thus, for deriving logical consequences, SUB1 is sound (with respect to the proof system without the consistent consequence rules, since then Γ is ignored). However, it is not sound for deriving consistent consequences. Consider the following example.

Example 4.0.1. Take the following BES.

$$(\mu\langle(X = A)(Y = B)\rangle)(\nu\langle(A = A)\rangle)(\mu\langle(B = B)\rangle)$$

In this BES, B could never be a consistent consequence of A, since A and B have different ranks. (In fact, the solution of A is *true* while the solution of B is *false*, thus if $A \lessdot B$ then we would have a counter example for Theorem 3.2.6.) However, by adding SUB1 to the proof system in Table 4.1, it is possible to build a proof tree with $\emptyset \vdash A \subset B$ as the root as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X \subset Y \vdash X \subset Y}\ \text{CNT}}{X \subset Y \vdash A \subset B}\ \text{SUB}}{\vdash X \subset Y}\ \text{CC}}{\vdash A \subset B}\ \text{SUB}$$

The problem with SUB1 is related to the fact that SUB1 manipulates the right-hand side of the $\vdash$ in the tree, without taking the left-hand side (or *context*) into consideration. To resolve this problem, one could redefine SUB1 such that it does behave appropriately, for example by also applying the substitution at the left-hand side of $\vdash$. However, then the rule cannot be used in the same way anymore to prove the completeness of the proof system.

We mentioned that the substitution rule is usually derivable from the other rules. Thus, one could imagine that this rule is not required for completeness of the system (at least, it is not required for completeness of the system with respect to logical consequence). In Chapter 5 we will prove that this is in fact the case; we will prove that the proof system as shown in Table 4.1 is in fact sound and complete for deriving consistent consequences on bound variables, by proving the following claim.

Claim For any BES $\mathcal{E}$, and for any propositional variables $X, Y \in \mathbf{bnd}$, $\emptyset \vdash X \subset Y$ is derivable *iff* $X \lessdot Y$.

4.0.1 Proof outline

If we can prove that the previous claim is true, then we would know that the proof system is sound and complete for deriving consistent consequences between bound variables; i.e. we can build a proof tree with $\vdash X \subset Y$ as the root iff Y is a consistent consequence of X.

To prove the claim, we first define a notion of relativization of consistent consequence relations. Similar to logical consequence, which can be defined relative to some relation on propositional variables, we can define a notion of relative consistent consequence, which takes the consistent consequence relation relative to some other relation (defined at the start of Chapter 5). This definition is such that, if we have some relation which is a consistent consequence relation relative to the empty relation, then this relation is a 'normal' consistent consequence relation.

We then lift this notion of relative consistent consequence on propositional variables to a notion of relative consistent consequence on propositional formulas. This notion is such that, for propositional variables X, Y, if propositional formula Y is a consistent consequence of propositional formula X relative to the empty relation, then Y is a consistent consequence of X.

In Section 5.1, we prove the soundness of the proof system (if we can build a proof tree with $\vdash X \subset Y$ as the root then Y is a consistent consequence of X). This is done by first showing that the proof system is sound for deriving relative consistent consequence between propositional formulas. The soundness follows then from the fact that consistent consequence between propositional formulas relative to the empty relation, if the propositional formulas are single variables, equates to normal consistent consequence between the variables.

Logical consequence lies at the basis of consistent consequence, and completeness of the proof system for logical consequence will be a useful tool for proving the completeness for consistent consequence. Therefore, before proving the completeness of the proof system for consistent consequence, we will prove that the proof system is complete for logical consequence in Section 5.2. To prove completeness for logical consequence, we will show how we can rewrite a propositional formula to an equivalent DNF in the proof system. Next we show that, if there is a logical consequence between a propositional formula in DNF and some other propositional formula, then we can build a proof tree with these propositional formulas in the root. The combination of these two gives us completeness of the proof system for logical consequence.

Finally, in Section 5.3 we will tackle completeness of the proof system for consistent consequence. We start with completeness of the proof system for consistent consequences relative to some relation Γ, which relates only bound variables from a BES. The intuition behind the proof is that we can build a consistent consequence relation at the left hand side of $\vdash$ in the proof system, which relates the variables at the root. We will prove this by induction on the number variables bound in the BES, but not related at the left hand side of $\vdash$. Since a consistent consequence relation relative to the empty relation is also a consistent consequence relation, we can then easily show that the proof system is also complete for deriving consistent consequences.

While proving the soundness and completeness properties of this proof system, we will introduce a few auxiliary and derived rules. For reference, a complete overview of these rules can be found in Appendix A.

Table 4.1: Proof system for consistent consequences on BES $\mathcal{E}$; α, β and γ are arbitrary propositional formulas; X, Y are propositional variables; Γ is a an arbitrary set of statements of the form $C \subset D$, where C and D are propositional variables

Axioms:

rules of the form $\dfrac{}{\Gamma \vdash A}$ where A ranges over:

AS1	$\alpha \wedge (\beta \wedge \gamma) \subset (\alpha \wedge \beta) \wedge \gamma$	DS1	$\alpha \vee (\beta \wedge \gamma) \subset (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
AS2	$(\alpha \wedge \beta) \wedge \gamma \subset \alpha \wedge (\beta \wedge \gamma)$	DS2	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma) \subset \alpha \vee (\beta \wedge \gamma)$
AS3	$\alpha \vee (\beta \vee \gamma) \subset (\alpha \vee \beta) \vee \gamma$	DS3	$\alpha \wedge (\beta \vee \gamma) \subset (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
AS4	$(\alpha \vee \beta) \vee \gamma \subset \alpha \vee (\beta \vee \gamma)$	DS4	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \subset \alpha \wedge (\beta \vee \gamma)$
COM1	$\alpha \wedge \beta \subset \beta \wedge \alpha$	AB1	$\alpha \vee (\alpha \wedge \beta) \subset \alpha$
COM2	$\alpha \vee \beta \subset \beta \vee \alpha$	AB2	$\alpha \subset \alpha \vee (\alpha \wedge \beta)$
ID1	$\alpha \subset \alpha \wedge \alpha$	ID2	$\alpha \vee \alpha \subset \alpha$
SUP	$\alpha \subset \alpha \vee \beta$	INF	$\alpha \wedge \beta \subset \alpha$
TOP	$\alpha \subset \alpha \wedge \top$	BOT	$\alpha \vee \bot \subset \alpha$

Logic rules:

TRA $$\frac{\Gamma \vdash \alpha \subset \beta \qquad \Gamma \vdash \beta \subset \gamma}{\Gamma \vdash \alpha \subset \gamma}$$ REF $$\frac{}{\Gamma \vdash \alpha \subset \alpha}$$

CTX $$\frac{\Gamma \vdash \alpha \subset \beta}{\Gamma \vdash \gamma[X := \alpha] \subset \gamma[X := \beta]}$$

Consistent Consequence rules:

CC $$\frac{\Gamma, X \subset Y \vdash f_X \subset f_Y}{\Gamma \vdash X \subset Y}\ \mathbf{rank}(X) = \mathbf{rank}(Y) \text{ and } X, Y \in \mathbf{bnd}_{\mathcal{E}}$$

CNT $$\frac{}{\Gamma \vdash X \subset Y}\ (X \subset Y \in \Gamma)$$

4.1 Some examples

In Example 3.1.6, we promised that we would show how the proof system can be used for deriving consistent consequences syntactically. As an appetizer for the next chapter, we will show how to construct the proof trees promised in this example.

Example 4.1.1. As a reminder, the BES we are considering is the following:

$$\begin{gathered}(\mu\langle(X_0 = X_4)(X_1 = X_0 \vee X_2)(X_2 = X_4 \vee X_5)(X_3 = X_2)(X_4 = X_4)\rangle)\\ (\nu\langle(X_5 = X_5)\rangle)\end{gathered}$$

The following proof trees derive (in order), $X_0 \lessdot X_2$, $X_1 \lessdot X_3$, $X_3 \lessdot X_1$ and $X_4 \lessdot X_0$.

$$\dfrac{\dfrac{}{X_0 \subset X_2 \vdash X_4 \subset X_4 \vee X_5}\,\text{SUP}}{\vdash X_0 \subset X_2}\,\text{CC}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X_1 \subset X_3 \vdash X_0 \subset X_2}\,\textit{Same as} \vdash X_0 \subset X_2}{X_1 \subset X_3 \vdash X_0 \vee X_2 \subset X_2 \vee X_2}\,\text{CTX} \qquad \dfrac{}{X_1 \subset X_3 \vdash X_2 \vee X_2 \subset X_2}\,\text{ID2}}{X_1 \subset X_3 \vdash X_0 \vee X_2 \subset X_2}\,\text{TRA}}{\vdash X_1 \subset X_3}\,\text{CC}$$

$$\dfrac{\dfrac{\dfrac{}{X_3 \subset X_1 \vdash X_2 \subset X_2 \vee X_0}\,\text{SUP} \qquad \dfrac{}{X_3 \subset X_1 \vdash X_2 \vee X_0 \subset X_0 \vee X_2}\,\text{COM2}}{X_3 \subset X_1 \vdash X_2 \subset X_0 \vee X_2}\,\text{TRA}}{\vdash X_3 \subset X_1}\,\text{CC}$$

$$\dfrac{\dfrac{}{X_4 \subset X_0 \vdash X_4 \subset X_4}\,\text{REF}}{\vdash X_4 \subset X_0}\,\text{CC}$$

The previous example does not use the rule CNT. However, this rule is truly required for completeness of the proof system, as shown in the following example.

Example 4.1.2. Consider the following BES.

$$(\mu\langle(X = A)(Y = B)\rangle)(\nu\langle(A = X)(B = Y)\rangle)$$

Then we have $X \lessdot Y$, and we can construct the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{X \subset Y, A \subset B \vdash X \subset Y}\,\text{CNT}}{X \subset Y \vdash A \subset B}\,\text{CC}}{\vdash X \subset Y}\,\text{CC}$$

Chapter 5

Proof system consistency

With relative consequence, we take the standard definition of logical consequence relative to some relation on propositional variables. We can do something similar with the notion of consistent consequence. We say that a relation R is a consistent consequence relation relative to another relation Γ, if it only relates variables of the same rank, and the right-hand sides of related bound variables are consequences relative to the union of R and Γ. Thus, the relative relation Γ relaxes only the second condition of the notion of consistent consequence; Γ need not relate variables of equal rank.

Definition 5.0.1. Let $\mathcal{E}$ be a BES and let R, Γ be relations on propositional variables. R is a *consistent consequence on $\mathcal{E}$ relative to* Γ if, for all X, Y such that $X \; R \; Y$, we have:

1. $\mathbf{rank}(X) = \mathbf{rank}(Y)$
2. if $X, Y \in \mathbf{bnd}$ then $f_X \overset{R\cup\Gamma}{\Longrightarrow} f_Y$

Given this notion of relative consequence relations, we can again define the largest such relation. Given a BES, and some relation Γ on propositional variables. For any propositional variables X, Y, if there exists a consistent consequence relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ relative to Γ such that $X \; (R \cup \Gamma \cup \mathfrak{I}) \; Y$, where $\mathfrak{I}$ denotes the identity relation relating all variables to themselves, then we say that Y is a consistent consequence of X relative to Γ, written $X \lessdot_\Gamma Y$.

The simplest example of the consistent consequence relation is the empty relation (denoted as $\emptyset$), which trivially satisfies all properties of the consistent consequence relation. Furthermore, the empty relation is also a consistent consequence relation relative to any relation.

Lemma 5.0.2. For any relation Γ on propositional variables and for any BES $\mathcal{E}$, $\emptyset$ is a consistent consequence relation on $\mathcal{E}$ relative to Γ.

We mentioned that the definition of relative consistent consequence would be such that, if the relation used for the relative part is the empty set, then the notions of relative consistent consequence and 'normal' consistent consequence coincide. This is shown by the following lemma shows this.

Lemma 5.0.3. For any BES $\mathcal{E}$ and relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ relating only bound variables, if R is a consistent consequence on $\mathcal{E}$ relative to $\emptyset$, then R is a consistent consequence on $\mathcal{E}$.

Proof. Let $\mathcal{E}$ be a BES and let R be a consistent consequence relative to $\emptyset$ on $\mathcal{E}$. R is a consistent consequence on $\mathcal{E}$ if, for all $(X, Y) \in R$, the ranks of X and Y are equal and, if both X and Y are bound in $\mathcal{E}$, then f_Y is a consequence of f_X relative to R. Both properties follow trivially from the definition of relative consistent consequence. □

Note that $\lessdot$ and $\lessdot_\emptyset$ are *not* the same. This is because $\lessdot_\emptyset$ also incorporates the identity relation. We do have $\lessdot \cup \mathfrak{I} = \lessdot_\emptyset$.

We can combine the definition of relative consequence and relative consistent consequence, to obtain a natural extension of relative consistent consequence on propositional variables to relative consistent consequence on propositional formulas as follows.

Definition 5.0.4. Let Γ be a relation on propositional variables and let f, g be propositional formulas. Let $\mathcal{E}$ be a BES. If there exists a consistent consequence relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ on $\mathcal{E}$ relative to Γ such that $f \overset{R \cup \Gamma}{\Longrightarrow} g$, then we say that g is a *relative consistent consequence of* f *under* Γ in $\mathcal{E}$.

For propositional formulas f, g, relation Γ and BES $\mathcal{E}$, if g is a relative consistent consequence of f under Γ in $\mathcal{E}$ and no confusion is possible, then we may abuse notation and write $f \overset{\lessdot_\Gamma}{\Longrightarrow} g$.

This definition of relative consistent consequence on propositional formulas does not coincide exactly with the definition of relative consistent consequence on propositional variables. In particular, given propositional variables X, Y and relation on propositional variables Γ and consistent consequence relation R relative to Γ, if we have $X \overset{R \cup \Gamma}{\Longrightarrow} Y$ then according to Lemma 2.2.19 we have $(X, Y) \in (R \cup \Gamma)^*$. However, we do not necessarily have $(X, Y) \in \lessdot_\Gamma$. Consider the following example.

Example 5.0.5. Let Γ be some relation on propositional variables. Furthermore, consider the following BES.

$$\mathcal{E} = (\mu\langle(X = X)\rangle)(\nu\langle(Y = Y)\rangle)(\mu\langle(Z = Z)\rangle)$$

Let $\Gamma = \{(X, Y), (Y, Z)\}$ be a relation on propositional variables. Then, $X \overset{\lessdot_\Gamma}{\Longrightarrow} Z$. However, for any consistent consequence relation R on $\mathcal{E}$ relative to Γ, R cannot relate X to Y, X to Z or Y to Z, since these are of differing ranks. Therefore, though we do have $(X, Z) \in (\lessdot_\Gamma)^*$, we do not have $X \ \lessdot_\Gamma \ Z$.

But we do not require these definitions to precisely coincide to use them to prove the proof system sound. We are mainly interested in the properties of the proof system with respect to consistent consequence, and not with respect to relative consistent consequence. Furthermore, Lemma 5.0.3 provides us with a correspondence between $\lessdot$ and $\lessdot_\emptyset$. It is sufficient to show that there is a correspondence between relative consistent consequence on propositional formulas (relative to $\emptyset$) and consistent consequence. This property is captured in the following Lemma.

Lemma 5.0.6. For any BES $\mathcal{E}$ and propositional variables $X, Y \in \mathbf{bnd}$, if $X \overset{\lessdot_\Gamma}{\Longrightarrow} Y$, then $X \lessdot Y$.

Proof. Let $\mathcal{E}$ be a BES, and let X, Y be variables bound in $\mathcal{E}$ such that $X \overset{\lessdot_\emptyset}{\Longrightarrow} Y$. We need to show that there exists a consistent consequence relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ such that $X \; R \; Y$.

Since $X \overset{\lessdot_\emptyset}{\Longrightarrow} Y$, there exists a consistent consequence relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ such that $X \overset{R \cup \emptyset}{\Longrightarrow} Y$. Take this R. Then, from Lemma 2.2.19 we know that $X \; (R \cup \emptyset)^* \; Y$. Thus, either $X = Y$ or $X \; (R)^+ \; Y$. If $X = Y$ then $\{(X, Y)\}$ is a consistent consequence relation on $\mathcal{E}$ relating X to Y, thus $X \lessdot Y$.

Otherwise $X \; R^+ \; Y$. In this case, if R^+ is a consistent consequence relation relating only bound variables, then $X \lessdot Y$.

Since R only relates bound variables, so does R^+. Furthermore, from Lemma 5.0.3 we know that, since R is a consistent consequence relation on $\mathcal{E}$ relative to $\emptyset$, R is a consistent consequence relation on $\mathcal{E}$.

Take variables $(X', Y') \in R^+$. Thus, we need to show that the ranks of X' and Y' are equal, and that the right hand side of Y' is a consequence of the right hand side of X' relative to R^+.

Since $(X', Y') \in R^+$, there exist variables $X_0, \cdots, X_n$ for some n such that $X_0 = X'$, $X_n = Y'$ and for all $i < n$ we have $X_i \; R \; X_{i+1}$. Since R is a consistent consequence relation, the ranks of $X_0, \cdots, X_n$ must all be equal, and thus so are the ranks of X' and Y'.

Now, take any environment θ consistent with R^+ such that $[\![f_X]\!]\theta = \mathit{true}$. We need to show that $[\![f_Y]\!]\theta = \mathit{true}$. Since θ is consistent with R^+, according to Lemma 2.2.9 it is also consistent with R.

Furthermore, since R is a consistent consequence relation, for all $i < n$ we have $f_{X_i} \overset{R}{\Rightarrow} f_{X_{i+1}}$. Thus, since θ is consistent with R and $[\![f_{X_0}]\!]\theta = \mathit{true}$, we have $[\![f_{X_i}]\!]\theta = \mathit{true}$ for all $i \leq n$, and thus also $[\![f_n]\!]\theta = \mathit{true} = [\![f_{Y'}]\!]\theta$. Therefore, $f_{X'} \overset{R^+}{\Longrightarrow} f_{Y'}$. $\square$

In the remainder of this chapter we will prove soundness and completeness of the proof system for deriving consistent consequences on a BES. Therefore, for the remainder of this chapter we will assume some arbitrary BES $\mathcal{E}$. Soundness and completeness is then formulated as follows. Given two bound variables X, Y, we can build a proof tree with $\emptyset \vdash X \subset Y$ as the root *iff* $X \lessdot^{\mathcal{E}} Y$.

5.1 Soundness

In this section, we will prove that the proof system is sound for deriving relative consistent consequences on propositional formulas. We will then use this property to prove soundness of the proof system for deriving consistent consequences between bound variables.

We start with proving that, if we can build a proof tree with two propositional formulas in the root and some relation Γ in the context (at the left hand side of $\vdash$), then the propositional formula on the right of $\subset$ is a consistent consequence of the propositional formula on the left of $\subset$ under Γ.

Lemma 5.1.1. For any propositional formulas f and g, and for any relation $\Gamma \subseteq \mathcal{X} \times \mathcal{X}$. If a tree has $\Gamma \vdash f \subset g$ as the root, then $f \overset{\lessdot_\Gamma}{\Longrightarrow} g$.

Proof. Given a proof tree with $\Gamma \vdash f \subset g$ as the root. We will prove that $f \overset{\lessdot_\Gamma}{\Longrightarrow} g$ by induction on the structure of the tree.

Axioms Soundness of the axioms and logic rules follows trivially from Lemma 2.2.5; for arbitrary relation R, if $f \Rightarrow g$ then we have $f \overset{R}{\Rightarrow} g$, and thus also $f \overset{\leq_\Gamma}{\Longrightarrow} g$.

CNT Take some relation Γ on propositional variables and propositional variables $(X, Y) \in \Gamma$. We need to show that there exists some relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ which is a consistent consequence relation relative to Γ such that $X \overset{R\cup\Gamma}{\Longrightarrow} Y$.

This is the case for the empty relation $R = \emptyset$. First, note that $R = \emptyset \subseteq \mathbf{bnd} \times \mathbf{bnd}$ (i.e. R only relates bound variables). Furthermore, from Lemma 5.0.2, it follows that R is a consistent consequence relative to Γ. Now, take any environment $\theta \in \Theta_{R\cup\Gamma}$ with $\theta(X) = \mathit{true}$. Since $(X, Y) \in \Gamma$, also $(X, Y) \in (R \cup \Gamma)$. Therefore, since $\theta \in \Theta_{R\cup\Gamma}$, also $\theta(Y) = \mathit{true}$, and thus $X \overset{R\cup\Gamma}{\Longrightarrow} Y$

CC We need to show that, for any relation on propositional variables Γ, and for any propositional variables $X, Y \in \mathbf{bnd}$ with equal rank, if we can build a proof tree with $\Gamma, (X \subset Y) \vdash f_X \subset f_Y$ as the root, then (propositional formula) Y is a relative consistent consequence of (propositional formula) X under Γ in $\mathcal{E}$.

Let X, Y and Γ be such that we can build a proof tree with this root. We need to find some relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ such that R is a consistent consequence relation relative to Γ and $X \overset{\Gamma\cup R}{\Longrightarrow} Y$.

From the induction hypothesis we know that f_Y is a relative consequence of f_X under $(\Gamma \cup \{(X, Y)\})$. Thus, there exists some relation $R' \subseteq \mathbf{bnd} \times \mathbf{bnd}$ which is a consistent consequence relation relative to $\Gamma \cup \{(X, Y)\}$, such that $f_X \xrightarrow{\Gamma\cup\{(X,Y)\}\cup R'} f_Y$. Let $R = R' \cup \{(X, Y)\}$. Then obviously $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ and, since $(X, Y) \in R$, also $(X, Y) \in (R \cup \Gamma)$ and thus trivially $X \overset{R\cup\Gamma}{\Longrightarrow} Y$.

What remains to show is that this R is a consistent consequence relation relative to Γ; Let X', Y' be any pair of variables related by R, then both X' and Y' are bound variables. We need to show that the ranks of X' and Y' are equal, and that $f_{X'} \overset{R\cup\Gamma}{\Longrightarrow} f_{Y'}$. Since R' only relates variables of equal rank, and the ranks of X and Y are equal, X' and Y' must also be of equal rank.

What remains to prove is that $f_{X'} \overset{R\cup\Gamma}{\Longrightarrow} f_{Y'}$. Since $(X', Y') \in R$, either $X' = X$ and $Y' = Y$ or $(X', Y') \in R'$. If $(X', Y') \in R'$ then $f_{X'} \overset{R\cup\Gamma}{\Longrightarrow} f_{Y'}$ follows from the fact that $R = R' \cup \{(X, Y)\}$ and R' is a consistent consequence relative to $\Gamma \cup \{(X, Y)\}$. If $X' = X$ and $Y' = Y$ then $f_{X'} \overset{R\cup\Gamma}{\Longrightarrow} f_{Y'}$ follows from our initial assumptions on X and Y. □

Our goal in this section was proving soundness of the proof tree for consistent consequence. Currently, we have soundness for relative consistent consequence on propositional formulas. Furthermore, from Lemma 5.0.6, if two propositional variables are consistent consequences relative to $\emptyset$ (using the definition of relative consistent consequence on propositional formulas), then they are related via a consistent consequence relation. These facts are sufficient to prove soundness of the proof system, as shown in the following theorem, which has been formalized in Coq in Lemma 6.7.2.

Theorem 5.1.2. For any bound variables X, Y, if we can construct a proof tree with $\emptyset \vdash X \subset Y$ as the root, then $X \lessdot Y$.

Proof. Assume we have a proof tree with $\emptyset \vdash X \subset Y$ as the root. Then, from Lemma 5.1.1 we know that $X \overset{\lessdot_\emptyset}{\Longrightarrow} Y$. Now, using Lemma 5.0.6, since X and Y are bound variables, we can conclude that $X \lessdot Y$.

□

5.2 Completeness for logical consequence

As mentioned before, the proof system in Table 4.1 has been derived from a complete and sound axiomatization for deriving logical consequences. These modifications invalidate the soundness properties of the the proof system for logical consequence (because of the rules CC and CNT), consider the following example.

Example 5.2.1. Consider the following BES: $\mu\langle(X = \top)(Y = \top)\rangle$.

Obviously we do not have $X \Rightarrow Y$. However, we can construct the following proof tree.

$$\cfrac{\cfrac{}{X \subset Y \vdash \top \subset \top}\,\text{REF}}{\vdash X \subset Y}\,\text{CC}$$

Thus, the proof system is unsound for deriving logical consequences between propositional formulas.

However, the completeness for deriving logical consequences has not been changed. Since at the basis of consistent consequence lies an implication relation, completeness of the proof system for logical consequence will be a valuable tool for deriving completeness of the proof system for consistent consequence. Since in the original proof system for logical consequence Γ does not participate, we will assume some arbitrary relation on propositional variables Γ as a context for the proof trees in this section.

Before we work on the completeness (for logical consequence) of the system, we start with deriving some additional rules allowing us to more easily construct the proof trees required in our proofs. The first states that if we have a conjunction in the consequent (the right hand side of the $\subset$ symbol), then it is sufficient to split the consequent and derive trees for each of the conjuncted formulas. This allows us to 'split' the consequent in the root of the tree into two separate trees.

Lemma 5.2.2. For any propositional formulas f, g_1, g_2, the following rule is derivable:

$$\cfrac{\Gamma \vdash f \subset g_1 \qquad \Gamma \vdash f \subset g_2}{\Gamma \vdash f \subset g_1 \wedge g_2}\,\text{cSPLIT}$$

Proof. We can construct the following tree:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f \subset f \wedge f}\,\text{ID1} \quad \dfrac{\Gamma \vdash f \subset g_2}{\Gamma \vdash f \wedge f \subset f \wedge g_2}\,\text{ID1}}{\Gamma \vdash f \subset f \wedge g_2}\,\text{TRA} \quad \dfrac{\Gamma \vdash f \subset g_1}{\Gamma \vdash f \wedge g_2 \subset g \wedge g_2}\,\text{CTX}}{\Gamma \vdash f \subset g_1 \wedge g_2}\,\text{TRA}$$

$\square$

Next we show that if we have a disjunction in the consequent, we can construct a proof tree if we can construct a proof tree for either of the disjuncted formulas. Thus, if we can construct a proof tree, then we can grow the consequent by placing it in disjunction with any propositional formula.

Lemma 5.2.3. For any propositional formulas f, g_1, g_2, the following rules are derivable:

$$\dfrac{\Gamma \vdash f \subset g_1}{\Gamma \vdash f \subset g_1 \vee g_2}\,\text{cGROW}_L \qquad \dfrac{\Gamma \vdash f \subset g_2}{\Gamma \vdash f \subset g_1 \vee g_2}\,\text{cGROW}_R$$

Proof. For cGROW$_L$ we can construct the following tree:

$$\dfrac{\dfrac{}{\Gamma \vdash f \subset f \vee g_2}\,\text{SUP} \quad \dfrac{\Gamma \vdash f \subset g_1}{\Gamma \vdash f \vee g_2 \subset g_1 \vee g_2}\,\text{CTX}}{\Gamma \vdash f \subset g_1 \vee g_2}\,\text{TRA}$$

The tree for cGROW$_R$ is similar to cGROW$_L$, using TRA and COM2 to rearrange the terms $\square$

The previous two rules manipulated the consequent of the root of the proof tree. We can do similar things for the antecedent (the right hand side of the $\subset$ symbol). First of all, if the antecedent consists of a disjunction of propositional formulas then we can construct a proof tree if we can construct proof trees for each of the disjuncted propositional formulas, allowing us to 'split' the antecedent in the root of the tree into two separate trees.

Lemma 5.2.4. For any propositional formulas f_1, f_2, g, the following rule is derivable:

$$\dfrac{\Gamma \vdash f_1 \subset g \qquad \Gamma \vdash f_2 \subset g}{\Gamma \vdash f_1 \vee f_2 \subset g}\,\text{aSPLIT}$$

Proof. We can construct the following tree:

$$\dfrac{\dfrac{\Gamma \vdash f_2 \subset g}{\Gamma \vdash f_1 \vee f_2 \subset f_1 \vee g}\,\text{CTX} \quad \dfrac{\dfrac{\Gamma \vdash f_1 \subset g}{\Gamma \vdash f_1 \vee g \subset g \vee g}\,\text{CTX} \quad \dfrac{}{\Gamma \vdash g \vee g \subset g}\,\text{ID2}}{\Gamma \vdash f_1 \vee g \subset g}\,\text{TRA}}{\Gamma \vdash f_1 \vee f_2 \subset g}\,\text{TRA}$$

$\square$

Finally, if the antecedent consists of a conjunction of propositional formulas then we can construct a proof tree if we can construct proof trees for one of the conjuncted propositional formulas, i.e. we can grow the antecedent by placing it in conjunction with any propositional formula.

Lemma 5.2.5. For any propositional formulas f_1, f_2, g, the following rules are derivable:

$$\frac{\Gamma \vdash f_1 \subset g}{\Gamma \vdash f_1 \wedge f_2 \subset g}\,\text{aGROW}_L \qquad \frac{\Gamma \vdash f_2 \subset g}{\Gamma \vdash f_1 \wedge f_2 \subset g}\,\text{aGROW}_R$$

Proof. We can construct the following tree for aGROW$_L$:

$$\frac{\dfrac{\Gamma \vdash f_1 \subset g}{\Gamma \vdash f_1 \wedge f_2 \subset g \wedge f_2}\,\text{CTX} \qquad \dfrac{}{\Gamma \vdash g \wedge f_2 \subset g}\,\text{INF}}{\Gamma \vdash f_1 \wedge f_2 \subset g}\,\text{TRA}$$

The tree for aGROW$_R$ is similar to aGROW$_L$, using TRA and COM1 to rearrange the terms. ☐

Using these rules, we can start working on proving the system complete for logical consequence. This will be done by first showing that we can rewrite any propositional formula to an equivalent propositional formula in DNF (see also Definition 2.1.2) in the proof system, and then showing that for all propositional formulas f and g such that g is a logical consequence of f, if f is in DNF, then we can derive a proof tree with $\Gamma \vdash f \subset g$ as the root.

To rewrite a propositional formula to an equivalent formula in DNF, we would like to have a function for obtaining a propositional formula in DNF which is equivalent to the input of the function. However, rewriting a propositional formula to an equivalent DNF is a fairly complex process, since we have to recursively distribute conjunctions over disjunctions, such that all disjunctions are at 'top-level', and all conjunctions are one level lower. Therefore, we first recursively define a function on DNFs, *dist*, which achieves this for a conjunction of two DNFs.

Definition 5.2.6. We recursively define a function *dist* on propositional formulas in DNF, for obtaining a DNF equivalent to the conjunction of its input.

$$\begin{aligned} \mathit{dist}(c_1)(c_2) &:= c_1 \wedge c_2 \\ \mathit{dist}(c_1)(f_1 \vee f_2) &:= (\mathit{dist}(c_1)(f_1)) \vee (\mathit{dist}(c_1)(f_2)) \\ \mathit{dist}(f_1 \vee f_2)(g) &:= \mathit{dist}(f_1 \vee g)(f_2 \vee g) \end{aligned}$$

Where f_1, f_2, g are DNF's and c_1, c_2 are clauses.

Using this distribute function, defining a function which given a propositional formula, produces an equivalent DNF, becomes easy.

Definition 5.2.7. We recursively define a function *DNF* on propositional formulas as follows.

$$\begin{aligned} \mathit{DNF}(\top) &:= \top \\ \mathit{DNF}(\bot) &:= \bot \\ \mathit{DNF}(X) &:= X \\ \mathit{DNF}(f_1 \wedge f_2) &:= \mathit{dist}(\mathit{DNF}(f_1))(\mathit{DNF}(f_2)) \\ \mathit{DNF}(f_1 \vee f_2) &:= \mathit{DNF}(f_1) \vee \mathit{DNF}(f_2) \end{aligned}$$

Where f_1, f_2 are propositional formulas.

The function DNF transforms a propositional formula into an equivalent DNF.

Lemma 5.2.8. Take a propositional formula f, then $DNF(f)$ is in DNF, and $DNF(f) \Leftrightarrow f$.

Proof. This property has been formalized in Coq, but we omit the proof here. We refer the interested reader to Coq Lemmas 6.3.6, 6.3.7 and 6.3.8. □

The following lemma shows that the rewrite steps of *dist* can be mimicked in the proof system, i.e. given a propositional formula which is the conjunction of two DNFs f, g, we can distribute the disjunctions across the conjunction to obtain the result of $dist(f)(g)$.

Lemma 5.2.9. For any DNF f, g, the following rule is derivable:

$$\frac{}{\Gamma \vdash f \wedge g \subset dist(f)(g)}\ \text{DIST}$$

Proof. Note: we will not explicitly mention when we resolve (parts of) the application of dist.

Take DNF's f, g. We need to show that it is possible to construct the required proof tree. We will prove this by induction on the structure of g:

$\top$ first, we use TRA and COM1 to switch f and g in the antecedent, then we apply induction on the structure of f:

- $\top$ We can build the proof tree using aGROW$_L$ and REF.
- $\bot$ We can build a proof tree using COM1 as follows.

$$\frac{}{\Gamma \vdash \bot \wedge \top \subset \top \wedge \bot}\ \text{COM1}$$

- Y We can build a proof tree using COM1.
- $f_1 \wedge f_2$ We can build a proof tree using COM1
- $f_1 \vee f_2$ In this case, we have the following induction hypotheses:
  IHf$_1$: We can derive a tree with $\Gamma \vdash \top \wedge f_1 \subset dist(f_1)(\top)$ as the root.
  IHf$_2$: We can derive a tree with $\Gamma \vdash \top \wedge f_2 \subset dist(f_2)(\top)$ as the root.

  We can construct the following proof tree, where the numbers are as indicated below.

$$\dfrac{\dfrac{}{(1)}\text{DS3} \qquad \dfrac{\dfrac{\dfrac{}{(2)}\text{IHf}_1}{(3)}\text{cGROW}_L \qquad \dfrac{\dfrac{}{(4)}\text{IHf}_2}{(5)}\text{cGROW}_R}{(6)}\text{aSPLIT}}{(7)}\text{TRA}$$

1. $\Gamma \vdash \top \wedge (f_1 \vee f_2) \subset (\top \wedge f_1) \vee (\top \wedge f_2)$
2. $\Gamma \vdash \top \wedge f_1 \subset dist(f_1)(\top)$
3. $\Gamma \vdash \top \wedge f_1 \subset (dist(f_1)(\top)) \vee (dist(f_2)(\top))$
4. $\Gamma \vdash \top \wedge f_2 \subset dist(f_2)(\top)$
5. $\Gamma \vdash \top \wedge f_2 \subset (dist(f_1)(\top)) \vee (dist(f_2)(\top))$
6. $\Gamma \vdash (\top \wedge f_1) \vee (\top \wedge f_2) \subset (dist(f_1)(\top)) \vee (dist(f_2)(\top))$

7. $\Gamma \vdash \top \wedge (f_1 \vee f_2) \subset (\mathit{dist}(f_1)(\top)) \vee (\mathit{dist}(f_2)(\top))$

$\bot$, X, $g_1 \wedge g_2$ Similar to the case where g is $\top$, except for the 'moment' of applying REF

$g_1 \vee g_2$ We have the following induction hypotheses:
IHg$_1$: for all DNFs f, we can construct a proof tree for $\Gamma \vdash f \wedge g_1 \subset \mathit{dist}(f)(g_1)$
IHg$_2$: for all DNFs f, we can construct a proof tree for $\Gamma \vdash f \wedge g_2 \subset \mathit{dist}(f)(g_2)$

We first make the following derivation, where the numbers are as indicated below:

$$\dfrac{\dfrac{}{(1)}\,\text{DS3} \qquad \dfrac{(2) \qquad (3)}{(4)}\,\text{aSPLIT}}{(5)}\,\text{TRA}$$

1. $\Gamma \vdash f \wedge (g_1 \vee g_2) \subset (f \wedge g_1) \vee (f \wedge g_2)$
2. $\Gamma \vdash f \wedge g_1 \subset \mathit{dist}(f)(g_1 \vee g_2)$
3. $\Gamma \vdash f \wedge g_2 \subset \mathit{dist}(f)(g_1 \vee g_2)$
4. $\Gamma \vdash (f \wedge g_1) \vee (f \wedge g_2) \subset \mathit{dist}(f)(g_1 \vee g_2)$
5. $\Gamma \vdash f \wedge (g_1 \vee g_2) \subset \mathit{dist}(f)(g_1 \vee g_2)$

We now still have to show that we can derive proof trees for $\Gamma \vdash f \wedge g_1 \subset \mathit{dist}(f)(g_1 \vee g_2)$ and $\Gamma \vdash f \wedge g_2 \subset \mathit{dist}(f)(g_1 \vee g_2)$ ((2) resp. (4)).

Constructing the proof tree for both cases is symmetrical up to some switching of cGROW$_L$ and cGROW$_R$, and switching IHg$_1$ for IHg$_2$ so we will only show how to construct the proof tree for (2). We do this via induction on the structure of f:

$\top$, $\bot$, Y, $f_1 \wedge f_2$ Can be done by using cGROW$_L$ and induction hypothesis IHg$_1$, as demonstrated below for $\top$:

$$\dfrac{\dfrac{}{\Gamma \vdash \top \wedge g_1 \subset \mathit{dist}(\top)(g_1)}\,\text{IHg}_1}{\Gamma \vdash \top \wedge g_1 \subset \mathit{dist}(\top)(g_1 \vee g_2)}\,\text{cGROW}_L$$

$f_1 \vee f_2$ We have the following induction hypotheses:
IHf$_1$:$\Gamma \vdash f_1 \wedge g_1 \subset \mathit{dist}(f_1)(g_1 \vee g_2)$ is derivable
IHf$_2$:$\Gamma \vdash f_2 \wedge g_1 \subset \mathit{dist}(f_2)(g_1 \vee g_2)$ is derivable

We start with making the following derivation, where the numbers are as indicated below:

$$\dfrac{\dfrac{\dfrac{}{(1)}\,\text{COM1} \qquad \dfrac{}{(2)}\,\text{DS3}}{(3)}\,\text{TRA} \qquad \dfrac{(4) \qquad (5)}{(6)}\,\text{aSPLIT}}{(7)}\,\text{TRA}$$

1. $\Gamma \vdash (f_1 \vee f_2) \wedge g_1 \subset g_1 \wedge (f_1 \vee f_2)$
2. $\Gamma \vdash g_1 \wedge (f_1 \vee f_2) \subset (g_1 \wedge f_1) \vee (g_1 \wedge f_2)$

3. $\Gamma \vdash (f_1 \vee f_2) \wedge g_1 \subset (g_1 \wedge f_1) \vee (g_1 \wedge f_2)$
4. $\Gamma \vdash g_1 \wedge f_1 \subset \mathit{dist}(f_1 \vee f_2)(g_1 \vee g_2)$
5. $\Gamma \vdash g_1 \wedge f_2 \subset \mathit{dist}(f_1 \vee f_2)(g_1 \vee g_2)$
6. $\Gamma \vdash (g_1 \wedge f_1) \vee (g_1 \wedge f_2) \subset \mathit{dist}(f_1 \vee f_2)(g_1 \vee g_2)$
7. $\Gamma \vdash f_1 \vee f_2 \wedge g_1 \subset \mathit{dist}(f_1 \vee f_2)(g_1 \vee g_2)$

We now still have to construct proof trees for $\Gamma \vdash g_1 \wedge f_1 \subset (f_1 \vee f_2)\mathit{dist}(g_1 \vee g_2)$ and $\Gamma \vdash g_1 \wedge f_2 \subset (f_1 \vee f_2)\mathit{dist}(g_1 \vee g_2)$ ((4) resp. (5)). The cases are symmetrical, up to a switching of cGROW$_L$ with cGROW$_R$ and IHf$_1$ with IHf$_2$, so we will only show the proof tree for $\Gamma \vdash g_1 \wedge f_1 \subset (f_1 \vee f_2)\mathit{dist}(g_1 \vee g_2)$. We can build the following proof tree, where the numbers are as indicated below:

$$\dfrac{\dfrac{\dfrac{}{(1)}\,\text{COM1} \qquad \dfrac{}{(2)}\,\text{IHf}_1}{(3)}\,\text{TRA}}{(4)}\,\text{cGROW}_L$$

1. $\Gamma \vdash g_1 \wedge f_1 \subset f_1 \wedge g_1$
2. $\Gamma \vdash f_1 \wedge g_1 \subset \mathit{dist}(f_1)(g_1 \vee g_2)$
3. $\Gamma \vdash g_1 \wedge f_1 \subset \mathit{dist}(f_1)(g_1 \vee g_2)$
4. $\Gamma \vdash g_1 \wedge f_1 \subset \mathit{dist}(f_1 \vee f_2)(g_1 \vee g_2)$

□

Now that we can distribute conjunctions over disjunctions in a conjunction of two formulas in DNF to obtain a DNF equivalent to the conjunction of two DNF's, we will show that from any propositional formula f we can derive an equivalent DNF, namely the result of $\mathit{DNF}(f)$. In other words, we will prove that for any propositional formula f, we can construct a proof tree with $\Gamma \vdash f \subset \mathit{DNF}(f)$ as the root.

Lemma 5.2.10. For any propositional formula f, the following rule is derivable:

$$\dfrac{}{\Gamma \vdash f \subset \mathit{DNF}(f)}$$

Proof. Note: we will not explicitly mention when we resolve (parts of) the application of DNF.

Let f be any propositional formula. We will show that it is possible to build a proof tree by induction on the structure of f.

$\top$ We can build a proof tree using REF

$\bot$ We can build a proof tree using REF

X We can build a proof tree using REF

$f_1 \wedge f_2$ In this case, we have the following induction hypotheses:
IHf$_1$: we can build a proof tree for $\Gamma \vdash f_1 \subset \mathit{DNF}(f_1)$
IHf$_2$: we can build a proof tree for $\Gamma \vdash f_2 \subset \mathit{DNF}(f_2)$

We can build the following proof tree, where the numbers are as indicated below:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(1)}\,\text{IHf}_1}{(2)}\,\text{aGROW}_L \qquad \dfrac{\dfrac{}{(3)}\,\text{IHf}_2}{(4)}\,\text{aGROW}_R}{(5)}\,\text{cSPLIT} \qquad \dfrac{}{(6)}\,\text{Lemma 5.2.9}}{(7)}\,\text{TRA}$$

1. $\Gamma \vdash f_1 \subset \mathit{DNF}(f_1)$
2. $\Gamma \vdash f_1 \wedge f_2 \subset \mathit{DNF}(f_1)$
3. $\Gamma \vdash f_2 \subset \mathit{DNF}(f_2)$
4. $\Gamma \vdash f_1 \wedge f_2 \subset \mathit{DNF}(f_2)$
5. $\Gamma \vdash f_1 \wedge f_2 \subset \mathit{DNF}(f_1) \wedge \mathit{DNF}(f_2)$
6. $\Gamma \vdash \mathit{DNF}(f_1) \wedge \mathit{DNF}(f_2) \subset \mathit{dist}(\mathit{DNF}(f_1))(\mathit{DNF}(f_2))$
7. $\Gamma \vdash f_1 \wedge f_2 \subset \mathit{dist}(\mathit{DNF}(f_1))(\mathit{DNF}(f_2))$

$f_1 \vee f_2$ In this case, we have the following induction hypotheses:
IHf$_1$: we can build a proof tree for $\Gamma \vdash f_1 \subset \mathit{DNF}(f_1)$
IHf$_2$: we can build a proof tree for $\Gamma \vdash f_2 \subset \mathit{DNF}(f_2)$

We can build the following proof tree, where the numbers are as indicated below.

$$\dfrac{\dfrac{\dfrac{}{(1)}\,\text{IHf}_1}{(2)}\,\text{cGROW}_L \qquad \dfrac{\dfrac{}{(3)}\,\text{IHf}_2}{(4)}\,\text{cGROW}_R}{(5)}\,\text{aSPLIT}$$

1. $\Gamma \vdash f_1 \subset \mathit{DNF}(f_1)$
2. $\Gamma \vdash f_1 \subset \mathit{DNF}(f_1) \vee \mathit{DNF}(f_2)$
3. $\Gamma \vdash f_2 \subset \mathit{DNF}(f_2)$
4. $\Gamma \vdash f_2 \subset \mathit{DNF}(f_1) \vee \mathit{DNF}(f_2)$
5. $\Gamma \vdash f_1 \vee f_2 \subset \mathit{DNF}(f_1) \vee \mathit{DNF}(f_2)$

□

We can now prove that the proof system is complete for logical consequence, when the antecedent is in DNF.

Lemma 5.2.11. For any propositional formulas f, g with f a DNF such that $f \Rightarrow g$ the following rule is derivable:

$$\dfrac{}{\Gamma \vdash f \subset g}$$

Proof. Given propositional formulas g and DNF f, such that $f \Rightarrow g$. We will show that it is possible to construct a proof tree by induction on the structure of f:

$\perp$ In this case we can construct the following proof tree, where the numbers are as indicated below:

$$\dfrac{\dfrac{\dfrac{}{(1)}\,\text{SUP} \qquad \dfrac{}{(2)}\,\text{COM2}}{(3)}\,\text{TRA} \qquad \dfrac{}{(4)}\,\text{BOT}}{(5)}\,\text{TRA}$$

1. $\Gamma \vdash \bot \subset \bot \vee g$
2. $\Gamma \vdash \bot \vee g \subset g \vee \bot$
3. $\Gamma \vdash \bot \subset g \vee \bot$
4. $\Gamma \vdash g \vee \bot \subset g$
5. $\Gamma \vdash \bot \subset g$

$\top$ Now, $\forall \theta : [\![g]\!]\theta = \mathit{true}$, since if there is a θ for which $[\![g]\!]\theta = \mathit{false}$ then $\top \Rightarrow g$ does not hold. We continue with induction on the structure of g:

$\bot$ This contradicts with the assumption that $\forall \theta : [\![g]\!]\theta = \mathit{true}$

$\top$ We can build a proof tree using REF.

Y This contradicts with our assumption that $\forall \theta : [\![g]\!]\theta = \mathit{true}$ (consider the environment which assigns *false* to all variables).

$g_1 \wedge g_2$ In this case both $\top \Rightarrow g_1$ and $\top \Rightarrow g_2$, so we can build a proof tree using the induction hypotheses of g and rule cSPLIT.

$g_1 \vee g_2$ In this case, we have the following induction hypotheses:
IHg$_1$: If $\top \Rightarrow g_1$ then we can build a proof tree with $\Gamma \vdash \top \subset g_1$ as the root.
IHg$_2$: If $\top \Rightarrow g_2$ then we can build a proof tree with $\Gamma \vdash \top \subset g_2$ as the root.
In this case, from Lemma 2.1.9, we have that $f \Rightarrow g_1$ or $f \Rightarrow g_2$. Thus we can build the proof tree using the induction hypothesis IHg$_1$ resp. IHg$_2$ and cGROW$_L$ resp. cGROW$_R$.

X We will show that we can build a proof tree via induction on the structure of g:

$\bot$ This contradicts with the assumption that $X \Rightarrow g$ (consider any environment assigning *true* to X).

$\top$ We can build the following proof tree, where the numbers are as indicated below.

$$\dfrac{\dfrac{}{(1)}\,\text{TOP} \qquad \dfrac{\dfrac{}{(2)}\,\text{COM1} \qquad \dfrac{}{(3)}\,\text{INF}}{(4)}\,\text{TRA}}{(5)}\,\text{TRA}$$

1. $\Gamma \vdash X \subset X \wedge \top$
2. $\Gamma \vdash X \wedge \top \subset \top \wedge X$
3. $\Gamma \vdash \top \wedge X \subset \top$
4. $\Gamma \vdash X \wedge \top \subset \top$
5. $\Gamma \vdash X \subset \top$

Y Now, either $X = Y$ or $X \neq Y$. In the first case, we can build a proof tree using REFL. The second case contradicts with the assumption that $X \Rightarrow Y$: consider an environment θ assigning *false* to everything but X. Since $X \neq Y$, we have $\theta(Y) = \mathit{false}$, while $\theta(X) = \mathit{true}$, and thus $X \Rightarrow Y$ does not hold.

$g_1 \wedge g_2$ We can build a proof tree in the same way as in the case $\top \subset g_1 \wedge g_2$.

$g_1 \vee g_2$ In this case, we have the following induction hypotheses:
IHg$_1$: If $X \Rightarrow g_1$ then we can build a proof tree with $\Gamma \vdash X \subset g_1$ as the root.
IHg$_2$: If $X \Rightarrow g_2$ then we can build a proof tree with $\Gamma \vdash X \subset g_2$ as the root.
In this case, from Lemma 2.1.9, we have that $f \Rightarrow g_1$ or $f \Rightarrow g_2$, and we can build a proof tree in the same way as in the case $\top \subset g_1 \vee g_2$.

$f_1 \wedge f_2$ In this case, both f_1 and f_2 are clauses since f is in DNF. Furthermore, we have the following induction hypotheses:
IHf$_1$: For all propositional formulas g, if $f_1 \Rightarrow g$ then we can build a proof tree with $\Gamma \vdash f_1 \subset g$ as the root.
IHf$_2$: For all propositional formulas g, if $f_2 \Rightarrow g$ then we can build a proof tree with $\Gamma \vdash f_2 \subset g$ as the root.

We will continue via induction on the structure of g:

$\top$ We can build a proof tree similarly to the case $X \subset \top$

$\bot$ In this case, either $f_1 \Rightarrow \bot$ or $f_2 \Rightarrow \bot$. If $f_1 \Rightarrow \bot$ then we can build a proof tree using aGROW$_L$ and IHf$_1$. The other case is similar replacing aGROW$_L$ with aGROW$_R$ and IHf$_1$ with IHf$_2$.

X Since $(f_1 \wedge f_2) \Rightarrow X$ and both f_1 and f_2 are clauses, either $X \in \mathcal{V}(f_1 \wedge f_2)$, or $f_1 \wedge f_2$ is equivalent to $\bot$.
If $X \in \mathcal{V}(f_1 \wedge f_2)$ then either $X \in \mathcal{V}(f_1)$ or $X \in \mathcal{V}(f_2)$, thus either $f_1 \Rightarrow X$ or $f_2 \Rightarrow X$. Thus, we can build a proof tree using aGROW$_L$ and IHf$_1$, resp. aGROW$_R$ and IHf$_2$.
Otherwise, if $f_1 \wedge f_2$ is equivalent to $\bot$ then either f_1 is equivalent to $\bot$ or f_2 is equivalent to $\bot$, in which case $f_1 \Rightarrow X$ resp. $f_2 \Rightarrow X$. Thus, we can build a tree using aGROW$_L$ resp. aGROW$_R$ and IHf$_1$ resp. IHf$_2$.

$g_1 \wedge g_2$ In this case, we have the following induction hypotheses on g:
IHg$_1$: If $f_1 \wedge f_2 \Rightarrow g_1$ then we can build a proof tree for $\Gamma \vdash f_1 \wedge f_2 \subset g_1$
IHg$_2$: If $f_1 \wedge f_2 \Rightarrow g_2$ then we can build a proof tree for $\Gamma \vdash f_1 \wedge f_2 \subset g_2$
Furthermore, both $f_1 \wedge f_2 \Rightarrow g_1$ and $f_1 \wedge f_2 \Rightarrow g_2$ hold. Therefore, we can build a proof tree using cSPLIT, IHg$_1$ and IHg$_2$.

$g_1 \vee g_2$ In this case we have the following induction hypotheses on g:
IHg$_1$: If $f_1 \wedge f_2 \Rightarrow g_1$ then we can build a proof tree for $\Gamma \vdash f_1 \wedge f_2 \subset g_1$
IHg$_2$: If $f_1 \wedge f_2 \Rightarrow g_2$ then we can build a proof tree for $\Gamma \vdash f_1 \wedge f_2 \subset g_2$
In this case, from Lemma 2.1.9, we have that $f \Rightarrow g_1$ or $f \Rightarrow g_2$, and we can build a proof tree in the same way as in the case $\top \subset g_1 \vee g_2$.

$f_1 \vee f_2$ In this case, we have the same induction hypotheses as in the case for $f_1 \wedge f_2$. Furthermore, both $f_1 \Rightarrow g$ and $f_2 \Rightarrow g$. Therefore, we can build a proof tree using aSPLIT and the induction hypotheses. □

Finally, we will prove that the proof system is complete for logical consequence on arbitrary propositional formulas. This proposition has been formalized in Coq in Lemma 6.7.12.

Proposition 5.2.12. For any propositional formulas f, g such that $f \Rightarrow g$, the following rule is derivable

$$\frac{}{\Gamma \vdash f \subset g}$$

Proof. Take propositional formulas f, g such that $f \Rightarrow g$. From Lemma 5.2.8 we know that $f \Rightarrow \mathit{DNF}(f)$ and $\mathit{DNF}(f) \Rightarrow f$. Take an environment θ such that $[\![\mathit{DNF}]\!]\theta = \mathit{true}$. Then also $[\![f]\!] = \mathit{true}$, and thus $[\![g]\!] = \mathit{true}$. Therefore, $\mathit{DNF}(f) \Rightarrow g$. Also, from Lemma 5.2.8 we know that $\mathit{DNF}(f)$ is a DNF.

Therefore we can build a proof tree as follows:

$$\dfrac{\dfrac{}{\Gamma \vdash f \subset \mathit{DNF}(f)}\text{ Lemma 5.2.10} \qquad \dfrac{}{\Gamma \vdash \mathit{DNF}(f) \subset g}\text{ Lemma 5.2.11}}{\Gamma \vdash f \subset g}\text{ TRA}$$

□

5.3 Completeness

The goal of the proof system is to be able to derive consistent consequences between (bound) propositional variables. Soundness of the system provides us with the useful property that if we can build a proof tree with two bound propositional variables in the root, then we may conclude that one is a consistent consequence of the other. What we still need is completeness of the proof system with respect to consistent consequence. By the end of this section, we will prove that the proof system is correct (i.e. sound and complete) for deriving consistent consequences on bound variables.

However, first we derive another auxiliary rule, which corresponds with a form of completeness for relative consequences.

Lemma 5.3.1. For all relations $\Gamma \subseteq \mathcal{X} \times \mathcal{X}$, $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$, and for all propositional formulas f, g such that $f \overset{R}{\Rightarrow} g$, the following rule is derivable:

$$\frac{\{\Gamma \vdash X \subset Y \mid (X, Y) \in R\}}{\Gamma \vdash f \subset g}$$

Proof. Take two propositional formulas f, g and relations on propositional variables Γ and $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ (i.e. R only relates bound variables) such that $f \overset{R}{\Rightarrow} g$. We need to show that, if we can build proof trees for all variables $(X, Y) \in R$ with $\Gamma \vdash X \subset Y$ as the root, then we can also build a proof tree with $\Gamma \vdash f \subset g$ as the root.

First, observe that we can use Lemma 5.2.12 to transform f and g at the root into two propositional formulas equivalent to f and g in DNF. Thus, if we prove that we can derive the rule for formulas in DNF, then we also have derivability of the rule for any pair of propositional formulas, which are a relative consequence of each other.

Let Γ be a relation on propositional variables, $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ and let f, g be two propositional formulas in DNF such that $f \overset{R}{\Rightarrow} g$. Furthermore, assume that for all propositional variables $(X, Y) \in R$, we can build a proof tree with $\Gamma \vdash X \subset Y$ as the root. We will prove the derivability of $\Gamma \vdash f \subset g$ by induction on the structure of f.

$\top$ In this case, g must be equivalent to $\top$ (consider the environment θ assigning *false* to all variables, this environment is consistent with R and $[\![\top]\!]\theta = \mathit{true}$, thus $[\![g]\!]\theta$ must also be *true*, which is only the case if g is equivalent to $\top$). Therefore, $\top \Rightarrow g$ and thus we can use completeness for logical consequence to build the proof tree.

$\bot$ In this case we can build the following proof tree, where the numbers are as indicated below:

$$\dfrac{\dfrac{\dfrac{}{(1)}\,\text{SUP} \qquad \dfrac{}{(2)}\,\text{COM2}}{(3)}\,\text{TRA} \qquad \dfrac{}{(4)}\,\text{BOT}}{(5)}\,\text{TRA}$$

1. $\Gamma \vdash \bot \subset \bot \vee g$
2. $\Gamma \vdash \bot \vee g \subset g \vee \bot$
3. $\Gamma \vdash \bot \subset g \vee \bot$
4. $\Gamma \vdash g \vee \bot \subset g$
5. $\Gamma \vdash \bot \subset g$

X In this case we show how to build a proof tree via induction on the structure of g:

$\top$ In this case we can build the following proof tree, where the numbers are as indicated below:

$$\dfrac{\dfrac{}{(1)}\,\text{TOP} \qquad \dfrac{\dfrac{}{(2)}\,\text{COM1} \qquad \dfrac{}{(3)}\,\text{INF}}{(4)}\,\text{TRA}}{(5)}\,\text{TRA}$$

1. $\Gamma \vdash \top \subset \top \wedge X$
2. $\Gamma \vdash \top \wedge X \subset X \wedge \top$
3. $\Gamma \vdash X \wedge \top \subset X$
4. $\Gamma \vdash \top \wedge X \subset X$
5. $\Gamma \vdash \top \subset X$

$\bot$ This contradicts our assumption that $X \overset{R}{\Rightarrow} g$, since the environment assigning *true* to all variables is consistent with R, and this environment assigns *false* to g.

Y In this case, since $X \overset{R}{\Rightarrow} Y$, from Lemma 2.2.19 we know that $X\ R^*\ Y$ thus either $X = Y$ or $X\ R^+\ Y$.

In the first case, we can build a proof tree using REF. Otherwise, since we can build a proof tree for all variables $(A, B) \in R$ with $\Gamma \vdash A \subset B$ as the root, using TRA we can also build proof trees for all variables $(A, B) \in R^+$, including (X, Y) (showing this is straightforward using induction on the number of variables required such that B is reachable from A through R).

$g_1 \wedge g_2$ If $g_1 \wedge g_2$ is equivalent to $\top$ then we can use the completeness of the system for logical consequence to build a proof tree. Otherwise, since $X \overset{R}{\Rightarrow} g_1 \wedge g_2$, for all variables $Y \in \mathcal{V}(g_1 \wedge g_2)$ we have $X \overset{R}{\Rightarrow} Y$, thus also $X\ R^*\ Y$. Therefore, we can repeatedly use cSPLIT and build proof trees for the resulting branches using a similar method as for the case $X \subset Y$.

$g_1 \vee g_2$ In this case, we have the following induction hypotheses:

IHg$_1$: If $X \overset{R}{\Rightarrow} g_1$ then we can build a proof tree with $\Gamma \vdash X \subset g_1$ as the root.

IHg$_2$: If $X \overset{R}{\Rightarrow} g_2$ then we can build a proof tree with $\Gamma \vdash X \subset g_2$ as the root.

Since $X \overset{R}{\Rightarrow} g_1 \vee g_2$, $\theta_{R,X}(X) = \mathit{true}$ and $\theta_{R,X}$ is consistent with R (2.2.11), either $[\![g_1]\!]\theta_{R,X} = \mathit{true}$ or $[\![g_2]\!]\theta_{R,X} = \mathit{true}$. Therefore, from Lemma 2.2.12, $X \overset{R}{\Rightarrow} g_1$ or $X \overset{R}{\Rightarrow} g_2$. Thus, we can build a proof tree using aSPLIT$_L$ resp. aSPLIT$_R$ and IHg$_1$ resp. IHg$_2$.

$f_1 \wedge f_2$ In this case, $f = f_1 \wedge f_2$ is a clause. We will prove a more general statement, namely that for any clause f and for any DNF g, if $f \overset{R}{\Rightarrow} g$ and we can build proof trees for all variables X, Y such that $X\ R\ Y$, then we can derive a proof tree for $\Gamma \vdash f \subset g$. We proceed with induction on the structure of g.

$\top$ In this case we can build a similar proof tree to the one built for the case $X \subset \top$.

$\bot$ In this case f is equivalent to $\bot$, thus we can build a proof tree from the completeness of the system for logical consequence.

Y If f is equivalent to $\bot$ then we can build a proof tree from the completeness of the system for logical consequence. Furthermore, f cannot be equivalent to $\top$, since then we would have $\top \overset{R}{\Rightarrow} Y$, and thus since θ_μ is consistent with R (from Lemma 2.2.2), we must also have $\theta_\mu(Y) = \mathit{true}$, which is not the case.

Assume that f is not equivalent to $\bot$ or $\top$. Since f is not equivalent to $\bot$, according to Lemma 2.2.14 we have $[\![f]\!]\theta_{R,f} = \mathit{true}$. Therefore, since according to Lemma 2.2.15 $\theta_{R,f}$ is consistent with R, we have $\theta_{R,f}(Y) = \mathit{true}$. Thus, from the definition of $\theta_{R,f}$, either there exists some $X \in \mathcal{V}(f)$ such that Y is reachable from X through R, or $Y \in \mathcal{V}(f)$.

In the first case, there exist variables $X_0, \cdots, X_n$ such that $X_0 = X$ and $X_n = Y$ and for all $i < n$ we have $X_i\ R\ X_{i+1}$. Thus, since we can build a proof tree for all variables $(A, B) \in R$ with $\Gamma \vdash A \subset B$ as the root, we can also build proof trees $\Gamma \vdash X_i \subset X_{i+1})$ for all $i < n$. Therefore, by repeatedly using TRA, we can construct a proof tree with $\Gamma \vdash X \subset Y$ as the root. From this, we can derive $\Gamma \vdash f \subset Y$ by repeatedly using aGROW$_L$, aGROW$_R$.

In the second case, we can derive $\Gamma \vdash X \subset Y$ using REF. From this, by repeatedly using aGROW$_L$ and aGROW$_R$, we can derive $\Gamma \vdash f \subset Y$.

$g_1 \wedge g_2$ In this case, we have the following induction hypotheses:

IHg$_1$: For all clauses f, if $f \overset{R}{\Rightarrow} g_1$ then we can build a proof tree with $\Gamma \vdash f \subset g_1$ in the root.

IHg$_2$: For all clauses f, if $f \overset{R}{\Rightarrow} g_2$ then we can build a proof tree with $\Gamma \vdash f \subset g_2$ in the root.

Furthermore, both $f \overset{R}{\Rightarrow} g_1$ and $f \overset{R}{\Rightarrow} g_2$ must hold. Therefore, we can use cSPLIT and the induction hypotheses to build the proof tree.

$g_1 \vee g_2$ In this case, we have the following induction hypotheses:

IHg$_1$: For all clauses f, if $f \overset{R}{\Rightarrow} g_1$ then we can build a proof tree with $\Gamma \vdash f \subset g_1$ in the root.

IHg$_2$: For all clauses f, if $f \overset{R}{\Rightarrow} g_2$ then we can build a proof tree with $\Gamma \vdash f \subset g_2$ in the root.

If f is equivalent to $\perp$ then we can construct a proof tree from the completeness for logical consequence. Otherwise, since f is not equivalent to $\perp$, we have $[\![f]\!]\theta_{R,f} = \mathit{true}$ from Lemma 2.2.14. Therefore, since according to 2.2.15 $\theta_{R,f}$ is consistent with R and $f \overset{R}{\Rightarrow} g$, we have $[\![g]\!]\theta_{R,f} = \mathit{true}$. Thus $[\![g_1]\!]\theta_{R,f} = \mathit{true}$ or $[\![g_2]\!]\theta_{R,f} = \mathit{true}$. The cases are symmetrical, so we will only consider the case $[\![g_1]\!]\theta_{R,f} = \mathit{true}$.

Since $[\![g_1]\!]\theta_{R,f} = \mathit{true}$, according to Lemma 2.2.17, also $f \overset{R}{\Rightarrow} g_1$. Thus we can build a proof tree using IHg$_1$ and aSPLIT$_L$.

$f_1 \vee f_2$ In this case, we have the following induction hypotheses:

IHf$_1$: For all DNF g such that $f_1 \overset{R}{\Rightarrow} g$ we can build a proof tree with $\Gamma \vdash f_1 \subset g$ as the root.

IHf$_2$: For all DNF g such that $f_2 \overset{R}{\Rightarrow} g$ we can build a proof tree with $\Gamma \vdash f_2 \subset g$ as the root.

Furthermore, we have both $f_1 \overset{R}{\Rightarrow} g$ and $f_2 \overset{R}{\Rightarrow} g$. Thus, we can build a proof tree using aSPLIT and the induction hypotheses. □

Next, we will prove completeness of the system for deriving consistent consequences on variables relative to some Γ which only relates bound variables. If this is the case, then it will be easy to prove that the proof system is complete for deriving consistent consequences on bound variables, since the empty relation also only relates bound variables, and Lemma 5.0.3 gives us a simple conversion from consistent consequence relative to the empty relation to general consistent consequence.

We mentioned that the intuition behind the proof for completeness is that we can build a consistent consequence relation on the left hand side of $\vdash$ which relates the variables in the root of the proof tree. Remember that $\lessdot$ is defined such that it only relates bound variables. We will show that we can grow the left hand side of $\vdash$ with pairs of bound variables, not currently occurring in Γ, but which are bound in the BES. The number of times we can do this is finite, since at some point in time the left hand side of $\vdash$ will relate all bound variables to all bound variables. When this is the case, we are certain that $\lessdot$ is a subset of Γ. Thus, we will prove the completeness of the proof system for relative consistent consequence by induction on the number of pairs of bound variables not currently occurring in Γ.

Definition 5.3.2. Given a relation $\Gamma : \mathbf{bnd} \times \mathbf{bnd}$. The *inverse size of* Γ, written $\#(\Gamma)$, is the size of the relation relating all bound variables not related by Γ. The following recursive function calculates this number.

$$\begin{aligned} \#(\Gamma) &= 0 && \text{if } \Gamma = \mathbf{bnd}^2 \\ \#(\Gamma) &= \#(\Gamma \cup \{(X,Y)\}) + 1 && \text{for any } (X,Y) \in (\mathbf{bnd}^2 \setminus \Gamma) \text{ if } \Gamma \subset \mathbf{bnd}^2 \end{aligned}$$

We can now fairly simply prove completeness for deriving consistent consequences on propositional variables, relative to relations relating only bound variables.

Lemma 5.3.3. For any relation $\Gamma \subseteq \mathbf{bnd}^2$, and for all propositional variables X, Y such that $X \lessdot_\Gamma Y$, we can derive the following rule:

$$\frac{}{\Gamma \vdash X \subset Y}$$

Proof. Take some X, Y such that $X \lessdot_\Gamma Y$. We will prove that we can construct a proof tree with $\Gamma \vdash X \subset Y$ as the root via induction on $\#(\Gamma)$:

$\#(\Gamma) = 0$ In this case $\Gamma = \mathbf{bnd} \times \mathbf{bnd}$. Since $X \lessdot_\Gamma Y$, either $X = Y$ or there exists a consistent consequence relation $R \subseteq \mathbf{bnd} \times \mathbf{bnd}$ relative to Γ such that $(X,Y) \in (\Gamma \cup R)$. In the first case we can build a proof tree using REF, otherwise, $R \subseteq \mathbf{bnd} \times \mathbf{bnd} = \Gamma$, and thus we can build a proof tree using CNT.

$\#\Gamma = x + 1$ The induction hypothesis tells us that for all Γ' with $\#\Gamma' = x$, and for all A, B with $A \lessdot_{\Gamma'} B$, we can build a proof tree with $\Gamma' \vdash A \subset B$ as the root.

We need to show that, for all X, Y with $X \lessdot_\Gamma Y$ we can build a proof tree with $\Gamma \vdash X \subset Y$ as the root. Take some X and Y such that $X \lessdot_\Gamma Y$. Then there exists a consistent consequence relation R relative to Γ such that $X\ (R \cup \Gamma \cup \mathfrak{I})\ Y$. If $(X,Y) \in \mathfrak{I}$ then $X = Y$ and we can build a proof tree using rule *REF*. If $(X,Y) \in \Gamma$ then we can build a proof tree using rule CNT.

If $X\ R\ Y$ and neither $X = Y$ nor $X\ \Gamma\ Y$, then the rank of X is equal to the rank of Y (since R is a relative consistent consequence relation relating only bound variables).

Furthermore, we have $f_X \overset{R \cup \Gamma}{\Longrightarrow} f_Y$. If for any propositional variables A, B such that $A\ (R \cup \Gamma)\ B$ we can derive a proof tree with $\Gamma, (X \subset Y) \vdash A \subset B$ as the root, then using 5.3.1, we can build the following proof tree.

$$\dfrac{\dfrac{\{\Gamma, X \subset Y \vdash A \subset B \mid (A, B) \in R \cup (\Gamma \cup (X, Y))\}}{\Gamma, X \subset Y \vdash f_X \subset f_Y}\text{Lemma 5.3.1}}{\Gamma \vdash X \subset Y}\text{CC}$$

What remains is to show that, for any propositional variables A, B such that $A\ (R \cup \Gamma)\ B$ we can build a proof tree with $\Gamma, (X \subset Y) \vdash A \subset B$ as the root.

Consider R. This relation is a consistent consequence relation relative to Γ, and thus R is obviously also a consistent consequence relation relative to $\Gamma \cup \{(X, Y)\}$. Now, take propositional variables $(A, B) \in (R \cup \Gamma)$. If $(A, B) \in \Gamma$ then we can build the proof tree using CNT. Otherwise, since $A\ R\ B$ and R is a consistent consequence relation relative to $\Gamma \cup \{(X, Y)\}$, and since $\#(\Gamma \cup \{(X, Y\}) = x$, we can use the induction hypothesis to build a proof tree with $\Gamma, X \subset Y \vdash A \subset B$ as the root. $\square$

Since we have show that the proof system is both sound and complete, the proof system satisfies it's requirements. This is captured by the following Theorem, which has been formalized in Coq in Lemma 6.7.16.

Theorem 5.3.4. For any $X, Y \in \mathbf{bnd}$, $\emptyset \vdash X \subset Y$ is derivable *iff* $X \lessdot Y$.

Proof.

$\Rightarrow$ From Lemma 5.1.2

$\Leftarrow$ From Lemma 5.0.2, if $X \lessdot Y$ then also $X \lessdot_{\emptyset} Y$. Furthermore, $\emptyset \subseteq \mathbf{bnd} \times \mathbf{bnd}$. Therefore we can build a proof tree according to Lemma 5.3.3. $\square$

Chapter 6

Consequences in Coq

We start this Chapter with some background information on Coq, and an explanation on some notation and some Coq peculiarities with which the reader may not be familiar. Then, we give some explanations on the contents of the Coq files which encode the theory from the previous 3 chapters [7].

6.1 Background

Coq is a Proof Assistant for a Logical Framework known as the Calculus of Inductive Constructions (CIC)[20]. CIC is a type theory, it is an extension of the Calculus of Constructions. The Calculus of Constructions was originally formulated by Thierry Coquand and Gérard Huet [24]. The extension was done by Christine Paulin. It can serve as both a typed programming language, and as a constructive foundation for mathematics. Coq allows the interactive construction of formal proofs, and also the manipulation of functional programs consistently with their specifications. It runs as a computer program on many architectures, and a wide variety of user interfaces is available [8].

Proving in Coq is based on the Curry-Howard isomorphism. The Curry-Howard isomorphism gives a correspondence between systems of formal logic as encountered in proof theory, and computational calculi as found in type theory. For instance, minimal propositional logic corresponds to simply typed λ-calculus, first-order logic corresponds to dependent types, etc.

The isomorphism has many aspects, even at the syntactic level: formulas correspond to types, proofs correspond to terms, provability corresponds to type inhabitation, proof normalization corresponds to term reduction, etc [17].

In Coq, we use this isomorphism to construct proofs. Proving corresponds with constructing a term of a type, which corresponds with the formula we wish to prove. For example, the following is a type.

$$\forall x.P(x)$$

It is also a proposition, and to prove this proposition, we need to find some object which has this type, i.e. we need to show that this type is inhabited. Constructing a proof in Coq is, in essence, telling Coq how to construct an object whose type is exactly what you wish to prove.

To trust Coq, we need to trust a number of things. We need to trust the theory behind Coq, the Coq kernel implementation, the Objective Caml compiler (since Coq was written in OCaml), the hardware on which Coq runs and any assumptions (axioms) made for creating the proofs [14].

The theory of Coq version 8.0 (which was used in this project) is proven to be consistent in Zermelo-Fraenkel set theory with inaccessible cardinals. Proofs of the consistency of subsystems of the theory of Coq can be found in literature.

For proofs in Coq to be correct, we need to trust that the implementation of the Coq kernel mirrors the theory behind Coq (CIC). The kernel was left intentionally small to limit the risk of conceptual or accidental implementation bugs.

The Coq kernel was written using the Objective Caml language but it uses only the most standard features (no object, no label ...), so that it is highly improbable that an Objective Caml bug breaks the consistency of Coq without breaking all other kinds of features of Coq or of other software compiled with Objective Caml.

In theory, if the hardware does not work properly, it can accidentally be the case that False becomes provable. But it is more likely the case that the whole Coq system will be unusable. However, if one feels the need, it is always possible to try compiling your proof using different computers.

Finally, any added axioms should be consistent with the theory behind Coq. Therefore, we tried to minimize the number of axioms we added to the Coq environment. The only axiom which was added, states that given a finite relation on propositional variables R, and two propositional variables X, Y, we can decide if $(X, Y) \in R$ is true or not. To decide this, since the relation is finite, we can simply check each of the pairs $(X', Y') \in R$, and see if (X, Y) is one of them.

6.2 The contents of this chapter

This chapter is designed to explain the contents of the Coq files which formalize the theory from this report, which may be found in [7]. Each of the Coq files starts with the following line, which tells the Coq compiler where to find the other files.

```
Add LoadPath "somePath".
```

If you wish to compile these files yourself, then this line should be changed to reflect the situation on your machine. Furthermore, the files should be compiled in the order in which they are listed in Table 6.1.

Table 6.1 states which section in this chapter corresponds with which Coq file. Each section starts with an overview of the definitions in the corresponding Coq file, as well as discussions on peculiarities and choices made during development. Then, the lemmas proven in the Coq file are presented and explained, in a slightly more readable format than the Coq code. However, since not every lemma states equally interesting properties, some of these descriptions have been moved to Appendix D. Please note that when explaining the lemmas, we will use $\Rightarrow$ and $\Leftrightarrow$ for *logical consequence* and *logical equivalence* as defined in Chapter 2. We will use $\rightarrow$ and $\leftrightarrow$ to indicate the logical implication and logical equivalence respectively. Furthermore, $\rightarrow$ will also be used to specify types (of

File name	**Section**
propForm.v	6.3
propForm_cons.v	6.3.1
DNF.v	6.3.2
cons_dec.v	6.3.3
propVar_relation.v	6.4
rel_cons.v	6.4.1
genBES.v	6.5
rel_reach.v	6.5.1
rel_ind.v	6.5.2
rel_cons_dec.v	6.5.3
BES.v	6.6
cc.v	6.6.1
rel_cc.v	6.6.2
prv_tree.v	6.7
sound.v	6.7.1
complete_cons.v	6.7.2
complete.v	6.7.3
sound_and_complete.v	6.7.4
cc_usefull.v	6.8

Table 6.1: The Coq files

mappings). Which interpretation is meant where is explained when it might not be obvious.

6.2.1 Exciting bits in the formalization

The theory behind most of the formalization in Coq has allready been discussed in the previous chapters. However, in some places the formalization was more complex than it would seem from the previous chapters.

In Section 6.3.2, we discuss DNFs and how these have been formalized in Coq. Because we inductively define the type of propositional formulas, Coq automatically generates induction principles which allow us to prove properties of propositional formulas by induction on the structure of propositional formulas. However, because we defined being in DNF as a property of propositional formulas, Coq does not automatically provide us with induction principles for proving by induction on the structure of DNFs. Therefore we have had to define and prove our own induction principles for DNFs (and also for clauses).

Furthermore, to define a function which, given some propositional formula, obtains an equivalent DNF, we separately defined a distribute function. Given two DNFs, this function returns a DNF which is equivalent to the conjunction of these DNFs. To define this function, we had to use a nested recursion function. This is because Coq expects a fixpoint to be *structurally decreasing* on one of its arguments, and it would not be possible to satisfy this requirement if we did not use nested recursion.

In Definition 5.3.2 we defined the inverse size of a relation. It is not a very complex definition, and by doing induction on the inverse size of a relation, we were able to prove the system complete for deriving consistent consequences between variables. However formalizing this definition in Coq has been a bit more complicated. We have had to define inductive types for being a relation which only relates bound variables, which we could relate to an 'inverse' constructive type for being a relation which only relates bound variables. On this inverse constructive type we could then define the inverse size of subset relations. This is discussed further in Section 6.5.2.

For formalizing the proof system, we found that we could effectively define the type of a proof tree as an inductive type with constructors for each of the rules in the proof system. This is explained in Secion 6.7.

Finally, as we argued in Section 2.3.1, to simplify the formalization of the relation between the semantics of BES and the consistent consequence relation, we have defined an alternate semantics for BES. The formalization of this semantics is presented in Section 6.8.

6.3 Propositional formulas

As variables, we will use the natural numbers, which we define as *propVar*. This way, we have access to an infinite set of variables. Moreover, equality on natural numbers is decidable in Coq. We can then inductively define propositional formulas as given in Definition 2.1.1.

```
Inductive propForm : Set :=
| top : propForm
| bot : propForm
| var : propVar -> propForm
| orp : propForm -> propForm -> propForm
| andp : propForm -> propForm -> propForm.
```

This notation should be read as follows. We inductively (*Inductive*) define the *Set* of propositional formulas as follows. *top* is a propositional formula; *bot* is a propositional formula; given a propositional variable X, we can construct a propositional formula by applying the constructor *var*; given two propositional formulas, we can construct another propositional formula by applying the constructor *orp*; given two propositional formulas, we can construct another propositional formula by applying the constructor *andp*. The following are some examples of propositional formulas (given some propositional variables X, Y, Z) as defined in Chapter 2 together with their Coq representations.

formula	Coq representation
$\top$	`top`
X	`var X`
$(X \wedge Y) \vee Z$	`orp (andp (var X) (var Y)) (var Z)`

To simplify writing propositional formulas, we introduced some infix notation in Coq for constructors *andp* ($\_ \wedge_p \_$) and *orp* ($\_ \vee_p \_$), which we will also be using in this document.

Furthermore, we defined a function *uses*: *propForm* $\rightarrow$ *propVar*, which determines if a propositional variable x occurs in a propositional formula f. Given a propositional formula, a function has been defined *mxUsed*: *propForm* $\rightarrow$ *propVar* which determines the largest natural number used to encode a propositional variable in some propositional formula. Finally, a function *replace*: *propForm* $\rightarrow$ *propVar* $\rightarrow$ *propForm* $\rightarrow$ *propForm* was defined. For propositional formulas f, g and propositional variable x, *replace*(f, x, g) returns the result of replacing every occurrence of x in f with g.

We define environments as mappings from propositional variables to the Booleans.

```
Definition environment := propVar -> bool.
```

The semantics of a propositional formula f is given in terms of an environment θ, as defined in Definition 2.1.5. We define a recursive function in Coq (hence the keyword *Fixpoint*, which indicates that we are defining a function using recursion) which calculates the semantics of a propositional formula in some environment as follows.

```
Fixpoint propForm_solution (f:propForm)(theta:environment):Prop:=
match f with
| a \/p b =>
       (propForm_solution a theta) \/ (propForm_solution b theta)
```

```
| a /\p b =>
       (propForm_solution a theta) /\ (propForm_solution b theta)
| var x => theta x
| T => True
| B => False
end.
```

In Coq, we write $[[f]]$*theta* for *propForm_solution f theta*, but in this document we use the notation $[\![f]\!]$*theta*.

We define a function *redefineEnvironment* : *environment* → *propVar* → *bool* such that, given an environment θ, propositional variable x and Boolean value s, *redefineEnvironment* θ x s returns an environment with x mapped to s and all other mappings as defined in θ. In Coq, and this document, we write $\theta[x := s]$ for *redefineEnvironment* θ x s.

```
Definition redefineEnvironment (theta:environment)(x:propVar)
                                          (s:bool):environment :=
fun (y:propVar)=>if beq_nat x y then s else theta y in newtheta.
```

Furthermore, the environments returning *true* and *false* for all variables are defined as *full_environment* resp. *empty_environment* in the Coq code. In this document, we referred to these as θ_ν resp. θ_μ. The environment returning *true* for a specific variable x and *false* otherwise is *environment_point*(x), and its inverse is *environment_max_point*(x).

```
Definition full_environment (x:propVar) : bool := true.
Definition empty_environment (x:propVar) : bool := false.
Definition environment_point (x y:propVar) :=
       if beq_nat x y then true else false.
Definition environment_max_point (x y:propVar) :=
       if beq_nat x y then false else true.
```

The union of two environments is defined as the environment returning *true* for a variable if either of the two environments returns *true* for this variable in Definition 2.1.3. The intersection of two environment is defined as the environment returning *true* for a variable if both of the two environments returns *true* for this variable. We will write $\theta_1 \& \theta_2$ for the intersection of environments θ_1 and θ_2.

```
Definition environment_union (theta1 theta2:environment)
                                                 (x:propVar) :=
       (theta1 x) || (theta2 x).
Definition environment_intersect (theta1 theta2:environment)
                                                 (x:propVar) :=
       (theta1 x) && (theta2 x).
```

Lemmas

Lemma 6.3.1. full_environment_maximal
For any propositional formula and environment, if the semantics of the propositional formula is *true* under this environment then the semantics of the propositional formula under the full environment is also *true*

$$\forall f, \forall \theta, [\![f]\!]\theta \to [\![f]\!]\theta_\nu$$

Lemma 6.3.2. empty_environment_minimal
For any propositional formula and environment, if the semantics of the propositional formula is *false* under this environment then the semantics of the propositional formula under the empty environment is also *false*.

$$\forall f, \forall \theta, \neg[\![f]\!]\theta \to \neg[\![f]\!]\theta_\mu$$

6.3.1 Logical consequence

Given two propositional formulas f, g, we say that g is a logical consequence of f if for all environments, the solution of f implies the solution of g, as defined in Definition 2.1.6. In this text we write $f \Rightarrow g$ instead of *propForm_cons f g*.

```
Definition propForm_cons (f g:propForm):Prop :=
  forall theta:environment, [[f]]theta -> [[g]]theta.
```

Given two propositional formulas f, g, we say that f is a logically equivalent to g if for all environments, the solution of f is equivalent to the solution of g. In this text we write $f \Leftrightarrow g$ instead of *propForm_eqv f g*.

```
Definition propForm_eqv (f g:propForm):Prop :=
  forall theta:environment, [[f]] theta <-> [[g]] theta.
```

Lemmas

Lemma 6.3.3. empty_true_eqv_top
If a propositional formula f evaluates to *true* under θ_μ then f is equivalent to $\top$

$$\forall f, [\![f]\!]\theta_\mu \to f \Leftrightarrow \top$$

6.3.2 DNF

Being a singleton is a property of propositional formulas. We define the function *singleton_propForm* to determine if a propositional formula is a singleton, which is consistent with Definition 2.1.2.

```
Definition singleton_propForm (f:propForm):Prop :=
match f with
| top => True
| bot => True
| var x => True
| _ => False
end.
```

Like with singletons, being a clause is a property of propositional formulas. We define a function *conj_propForm* which determines if a propositional formula is a clause, which is consistent with Definition 2.1.2. We will write *clause*(f) for the proposition *conj_propForm f*.

```
Fixpoint conj_propForm (f:propForm):Prop :=
match f with
| a /\p b => conj_propForm a /\ conj_propForm b
| _ => singleton_propForm f
end.
```

Coq provides us with standard methods for doing induction on the structure of propositional formulas. However, for induction on the structure of a clause we need to define our own induction principle. This is because being a clause is defined as a property of propositional formulas, in stead of being defined as some inductive type. In Coq, we have proven the following induction principle, which allows us to do induction on the structure of propositional formulas which are clauses.

```
Definition conj_ind : forall P:propForm -> Prop,
  P top ->
  P bot ->
  (forall x:propVar, P (var x)) ->
  (forall f g:propForm, conj_propForm f -> conj_propForm g ->
    P f -> P g -> P (f /\p g)) ->
  (forall f:propForm, conj_propForm f -> P f).
```

With *conj_*, we have defined the following. For any proposition P over propositional formulas, if P holds on $\top$; and P holds on $\bot$; and P holds on all formulas consisting of single propositional variables; and for all clauses f, g if P holds on both f and g then P also holds on the conjunction of f and g; then we can conclude that P holds on all clauses.

As with singletons and clauses, being in disjunctive normal form (DNF) is a property of propositional formulas. Like in Definition 2.1.2, we have inductively defined a function *DNF_propForm* which determines if a propositional formula is in DNF. In this document, we write *DNF f* to indicate that f is in DNF.

```
Fixpoint DNF_propForm (f:propForm):Prop :=
match f with
| f1 \/p f2 => (DNF_propForm f1 /\ DNF_propForm f2)
| _ => conj_propForm f
end.
```

We can also do induction on the structure of DNFs. However, like with clauses, we need to define (and proof correct) such an induction principle separately. The following is the induction principle we have proven in Coq, which allows us to do induction on the structure of DNFs.

```
Definition DNF_ind : forall P:propForm -> Prop,
  P top ->
  P bot ->
  (forall x:propVar, P (var x)) ->
  (forall f g:propForm, conj_propForm f -> conj_propForm g ->
    P f -> P g -> P (f /\p g)) ->
  (forall f g:propForm, DNF_propForm f -> DNF_propForm g ->
    P f -> P g -> P (f \/p g)) ->
  (forall f:propForm, DNF_propForm f -> P f).
```

To transform a propositional formula, we need to recursively distribute conjunctions over disjunctions. The function *distribute* takes two propositional formulas f, g, and returns a DNF equivalent to the conjunction of f and g. It corresponds with the function presented in Definition 5.2.6. In this text we will write $\mathit{dist}(f)(g)$ for *distribute f g*.

```
Fixpoint distribute (f:propForm) : propForm -> propForm :=
fix distribute1 (g:propForm) : propForm :=
match f with
| f1 \/p f2 => distribute f1 g \/p distribute f2 g
| _ => match g with
       | g1 \/p g2 => distribute1 g1 \/p distribute1 g2
       | _ => f /\p g
       end
end.
```

We define a function *toDNF*, which given a propositional formula, returns an equivalent formula in DNF. This function corresponds with the one from Definition 5.2.7. In this chapter, we will write $\mathit{toDNF}(f)$ for the result of applying *toDNF* to formula f, and $\mathit{DNF}(f)$ for the proposition that f is in DNF. This is different from the previous Chapters, where we wrote $\mathit{DNF}(f)$ for the result of applying *toDNF* to f.

```
Fixpoint toDNF (f:propForm):propForm :=
match f with
| f1 \/p f2 => toDNF f1 \/p toDNF f2
| f1 /\p f2 => distribute (toDNF f1) (toDNF f2)
| _ => f
end.
```

Lemmas

Lemma 6.3.4. distribute_DNF
Distribute called on two DNF formulas f, g returns a DNF formula.

$$\forall fg, \mathit{DNF} f \rightarrow \mathit{DNF} g \rightarrow \mathit{DNF}(\mathit{dist}(f)(g)).$$

Lemma 6.3.5. distribute_sound
Given two propositional formulas f, g and an environment θ, if $[\![f]\!]\theta = \mathit{true}$ and $[\![g]\!]\theta = \mathit{true}$ then the semantics of the result of distributing f over g under θ is also *true*.

$$\forall fg, \forall\theta, [\![f]\!]\theta \rightarrow [\![g]\!]\theta \rightarrow [\![\mathit{dist}(f)(g)]\!]\theta$$

Lemma 6.3.6. DNF_toDNF
For any propositional formula f, the result of transforming f using *to_DNF* is a DNF.

$$\forall f, \mathit{DNF}(\mathit{toDNF}(f))$$

Lemma 6.3.7. toDNF_cons
For any propositional formula f and any environment θ, the solution of $[\![toDNF f]\!]\theta$ implies the solution of $[\![f]\!]\theta$

$$\forall f, \forall\theta, [\![toDNF(f)]\!]\theta \rightarrow [\![f]\!]\theta$$

Lemma 6.3.8. cons_toDNF
For any propositional formula f and any environment θ, the solution of $[\![f]\!]\theta$ implies the solution of $[\![toDNF f]\!]\theta$

$$\forall f, \forall\theta, [\![f]\!]\theta \rightarrow [\![toDNF(f)]\!]\theta$$

6.3.3 Logical consequence decidability

Lemmas

Lemma 6.3.9. propForm_cons_propForm_decidable
For any propositional formulas f, g, either g is a logical consequence of f, or there exists an environment for which f evaluates to *true* and g evaluates to *false*

$$\forall f g, (f \Rightarrow g \vee \exists\theta, ([\![f]\!]\theta \wedge \neg[\![g]\!]\theta))$$

Lemma 6.3.10. propForm_eqv_decidable
For any propositional formulas f, g, either f is equivalent to g or not

$$\forall f g, f \Leftrightarrow g \vee \neg(f \Leftrightarrow g)$$

6.4 Relations on variables

We define a relation on propositional variables as any function mapping two propositional variables to propositions. Let R be a relation, and let x, y be propositional variables. Then $R\ x\ y$ is a proposition corresponding with $x\ R\ y$ and $(x, y) \in R$. Observe that with this formulation, membership in a relation is in Prop, and thus it is in general undecidable if a pair of variables is related by a relation.

```
Definition propVar_relation := propVar -> propVar -> Prop
```

The empty relation is the relation which does not relate any variables; i.e. any two variables x, y are not related by *empty_relation*. In this text, we write $\emptyset$ to indicate this relation.

```
Definition empty_relation (x y:propVar):Prop := False
```

The identity relation is the relation relating all variables to themselves, and only that. We refer to this relation as *id_rel*. In this text, we write $\mathfrak{I}$ to indicate the identity relation.

```
Definition id_rel := fun (x y:propVar) => x=y
```

We say that two relations R_1, R_2 are equivalent, *rel_eqv*, if for all variables x, y, $(x, y) \in R_1$ if and only if $(x, y) \in R_2$. We will write $R_1 \sim R_2$ to indicate that R_1 and R_2 are equivalent.

```
Definition rel_eqv (R1 R2:propVar_relation) :=
  forall x y:nat, R1 x y <-> R2 x y
```

Given two relations R_1, R_2, we define the union of these relations as the relation relating only and all variables related by either R_1 or R_2, i.e. $\{(a, b) \mid (a, b) \in R_1 \vee (a, b) \in R_2\}$. In this text we write $R_1 \cup R_2$ to indicate the union of R_1 and R_2.

```
Definition rel_union : propVar_relation ->
                    propVar_relation -> propVar_relation :=
let result (R1 R2 : propVar_relation) (x z:propVar) : Prop :=
  (R1 x z) \/ (R2 x z)
in result
```

Given two relations R_1, R_2, we define *rel_minus* $R_1\ R_2$ as the relation relating only and all variables related by R_1 except for those related by R_2, i.e. $\{(a, b) \mid (a, b) \in R_1(a, b) \notin R_2\}$. We will write $R_1 \setminus R_2$ to indicate *rel_minus* $R_1\ R_2$.

```
Definition rel_minus (R1 R2:propVar_relation):propVar_relation :=
fun (x y:propVar) => R1 x y /\ ~R2 x y
```

Given a pair of propositional variables x, y, we define the relation relating only x to y as *pair_relation* $x\ y$, i.e. $\{(x, y)\}$. We will write $\{(x, y)\}$ for *pair_relation*$(x\ y)$.

```
Definition pair_relation (x y:propVar) : propVar_relation :=
  let result (x' y':propVar) : Prop := (x=x') /\ (y=y') in result
```

Given a relation R and propositional variable a, *remove_var_left* R a is the relation resulting from removing from R all tuples (a, y). i.e. $\{(x, y) \mid (x, y) \in R \wedge x \neq a\}$. In this text we will write $R \setminus_L a$ to indicate *remove_var_left* R a. We define this relation as follows.

```
Definition remove_var_left (R:propVar_relation)(a:nat)(x y:nat)
                                                       : Prop :=
if (beq_nat a x) then False else R x y
```

Note: In Coq, if_then_else_ is notation for an operator from the Booleans to some other type. The type of $a = x$ is a proposition, and thus cannot be used in if-the-else statements. beq_nat is an operator on the natural numberss, which takes two natural numbers (here a, x) and returns true if $a = x$ and false otherwise. Therefore, we use beq_nat a x instead of $a = x$ in the if-the-else statement.

Given a relation R and propositional variable b, *remove_var_right* R b is the relation resulting from removing from R all tuples (x, b). i.e. $\{(x, y) \mid (x, y) \in R \wedge y \neq b\}$. In this text we will write $R \setminus_R b$ to indicate *remove_var_right* R b.

```
Definition remove_var_right (R:propVar_relation)(b:nat)(x y:nat)
                                                       : Prop :=
if (beq_nat b y) then False else R x y
```

Given a relation R and propositional variables a, b, *remPoint* R a b is the relation resulting from removing from R only the tuple (a, b). i.e. $\{(x, y) \mid (x, y) \in R \wedge x \neq a \wedge y \neq b\}$.

```
Definition remPoint (R:propVar_relation)(a b:nat)(x y:nat):Prop:=
if ((beq_nat a x)&&(beq_nat b y)) then False else R x y
```

6.4.1 Relative consequences

Given a relation on propositional variables and an environment, we say that the environment is *consistent* with the relation if, for all pairs of variables related by the relation, the interpretation of the variables under the environment is related via a logical consequence, as in Definition 2.2.1.

```
Definition consistent_environment (theta : environment)
                                         (R:propVar_relation) :=
forall x y: propVar, (R x y) -> (theta x -> theta y)
```

Given a relation on propositional variables and a pair of propositional formulas, we say that the propositional formulas are a *consequence* of one another relative to the relation if, for all environments consistent with the relation, the interpretation of the formulas under these environments are logical consequences, as in Definition 2.2.4.

```
Definition rel_cons (R:propVar_relation)(f g:propForm) :=
forall theta:environment, consistent_environment theta R ->
  [[f]]theta -> [[g]]theta.
```

Lemmas

Lemma 6.4.1. environment_union_consistent
For any relation on propositional variables. If any two environments are consistent with this relation, then so is the union of these environments.

$$\forall R, \forall \theta_1 \theta_2, \theta_1 \in \Theta_R \to \theta_2 \in \Theta_R \to (\theta_1 || \theta_2) \in \Theta_R$$

Lemma 6.4.2. full_environment_consistent
The full environment (the environment assigning *true* to all variables) is consistent with all relations on propositional variables.

$$\forall R, \theta_\nu \in \Theta_R$$

Lemma 6.4.3. empty_environment_consistent
The empty environment (the environment assigning *false* to all variables) is consistent with all relations on propositional variables.

$$\forall R, \theta_\mu \in \Theta_R$$

Lemma 6.4.4. consistent_environment_rel_union
If an environment is consistent with the union of two relations, then the environment is consistent with both relations.

$$\forall \theta, \forall R_1 R_2, \theta \in \Theta_{R_1 \cup R_2} \to (\theta \in \Theta_{R_1} \land \theta \in \Theta_{R_2})$$

Lemma 6.4.5. rel_union_consistent_environment
If an environment is consistent with two relations, then the environment is also consistent with the union of these relations.

$$\forall \theta, \forall R_1 R_2, (\theta \in \Theta_{R_1} \land \theta \in \Theta_{R_2}) \to \theta \in \Theta_{R_1 \cup R_2}$$

6.5 Boolean Equation Systems

We inductively define Boolean equations with constructor *bEqn*, consistent with Definition 2.3.1. The constructor takes a propositional variable and propositional formula to build a Boolean equation. We will write $(x = f)$ for Boolean equation *bEqn*$(x\ f)$.

```
Inductive booleanEquation :=
  bEqn : propVar -> propForm -> booleanEquation.
```

A block is a list of Boolean equations. Its block type is either μ or ν, which we capture using a Boolean (*false* resp. *true*). Furthermore, a block is never empty. Thus, a block is a triple of a list of Boolean equations, its block type and a proof that the list is non-empty, as defined in Definition 2.3.2.

```
Record block:=makeBlock{
  theBlock:> list booleanEquation;
  blockType: bool;
  non_empty: exists beqn:booleanEquation, In beqn theBlock
}.
```

If a list of blocks has alternating block types, then the list of blocks is a (not necessarily well-formed) BES. We define a function on lists of Boolean equations, which decides if the block types of the blocks in the list alternate.

```
Fixpoint alternating_list_block (theList:list block):Prop:=
match theList with
| nil => true
| b1::l => match l with
           | nil => true
           | b2::l' =>
                 (eqb (blockType b1) (negb (blockType b2)))
                                 /\ alternating_list_block l
           end
end.
```

A BES is defined as a list blocks of alternating types, as in Definition 2.3.4. Thus, it is a tuple of a list of blocks and a proof that the list of blocks has alternating block types.

```
Record genBES := make_genBES {
  the_genBES:> list block;
  block_alternates: alternating_list_block the_genBES
}.
```

We define a function *bnd_block* to determine if a propositional variable x occurs on the left-hand side of an equation in a block. We will write $x \in B$ to indicate that x is bound in block B.

```
Fixpoint bnd_block (E:list booleanEquation)(x:propVar):Prop:=
match E with
| nil => False
| cons (bEqn y _) E' => x=y \/ bnd_block E' x
end.
```

Using this function, we can define a function *bnd* which determines if a propositional variable x occurs on the left-hand side of one of the blocks in a list of blocks (or, a genBES). We will abuse notation and write $x \in E$ to indicate that x is bound in list of blocks E.

```
Fixpoint bnd (E:list block)(x:propVar) : Prop :=
match E with
| nil => False
| cons bl E' => bnd_block bl x \/ bnd E' x
end.
```

We define a function *bnd_cnt_block* to determine how often a propositional variable x occurs on the left hand side of an equation in block E. We will write $\#(x \in B)$ instead of *bnd_cnt_block* B x.

```
Fixpoint bnd_cnt_block (bl:list booleanEquation)(x:propVar):nat:=
match bl with
| nil => 0
| bEqn y _ :: bl' => if (beq_nat x y)
                     then S(bnd_cnt_block bl' x)
                     else bnd_cnt_block bl' x
end.
```

Using this function, we can define a function *bnd* which determines how often a propositional variable x occurs on the left hand side of one the blocks in a list of blocks (or, a genBES). We will abuse notation and write $\#(x \in E)$ instead of *bnd_cnt* E x.

```
Fixpoint bnd_cnt (E:list block)(x:propVar) : nat :=
match E with
| nil => 0
| cons bl E' => bnd_cnt_block bl x + bnd_cnt E' x
end.
```

Finally, we define a function *occ_block* to determine if a propositional variable occurs in the right-hand side of an equation in a block.

```
Fixpoint occ_block (E:list booleanEquation)(x:propVar) : Prop :=
match E with
| nil => False
| cons (bEqn _ f) E' => (uses f x) \/ (occ_block E' x)
end.
```

Which we use to define a function *occ* which determines if a propositional variable occurs in the right-hand side of an equation in a list of blocks (or, a genBES).

```
Fixpoint occ (E:list block)(x:propVar) : Prop:=
match E with
| nil => False
| cons bl E' => occ_block bl x \/ occ E' x
end.
```

6.5.1 Reachability on relations

Given a relation on propositional variables R and two lists of Boolean equations l_1, l_2, if R only relates variables bound in l_1 to variables bound in l_2 then we say that R only relates bound variables, which we name *relates_bound*. In this chapter we will write $R \subseteq l_1 \times l_2$ for *relates_bound* R l_1 l_2, and $R \subseteq l_1 \times l_2$, using the notation from previous chapters, corresponds exactly with $R \subseteq \mathbf{bnd}_{l_1} \times \mathbf{bnd}_{l_2}$.

```
Definition relates_bound (R:propVar_relation) (l1 l2:genBES):=
  (forall x y:nat, R x y -> (bnd l1 x /\ bnd l2 y))
```

Because we defined relations such that membership of a relation is in *Prop*, we cannot prove for arbitrary relation R and propositional variables X, Y, that $(X, Y) \in R$ (neither can we prove $(X, Y) \notin R$). However, this is decidable for the relations which only relate bound variables we are dealing with. Therefore we add this assumption to our Coq environment. For any relation R relating only variables bound in some BES to variables bound in another BES, and for any propositional variables x, y, it is decidable whether R relates x to y or not.

```
Axiom relates_decidable :
  forall R:propVar_relation, forall l1 l2:genBES,
    relates_bound R l1 l2 -> forall x y:nat, {R x y}+{~R x y}
```

For any pair of lists of Boolean equations l_1, l_2, the largest relation relating only variables bound in l_1 to variables bound in l_2 is called *max_rel*, in this chapter written as $l_1 \times l_2$.

```
Definition max_rel (l1 l2:genBES)(a b:nat):Prop:=
  (bnd l1 a /\ bnd l2 b)
```

Given a relation R, we say that a variable y is *reachable* from another variable x through R if we have some set of variables $x_0, \cdots, x_n$ for some n such that $x_0 = x$, $x_n = y$ and for all $i < n$, R relates x_i to x_{i+1}. We can inductively define this property as: if R relates x to y then y is reachable from x; if R relates x to some z, and y is reachable from z through R then y is reachable from x. As shown in Lemma 2.2.8, the relation induced by the reachable set of a relation R corresponds exactly with R^+.

```
Inductive reachable (R:propVar_relation)
                                 : propVar -> propVar -> Prop:=
| single_path (x y:propVar) : R x y -> reachable R x y
| step_path (x z:propVar)   : forall y:propVar, R x y ->
    reachable R y z -> reachable R x z
```

In Definitions 2.2.10 and 2.2.13, we defined minimal environments consistent with some relation R for propositional variables resp. formulas. To define these environments in Coq, we require that determining if a variable y is reachable from a variable x through some relation R is decidable. However, this is not in general decidable. But, it is sufficient if we take relations which only relate bound variables (for which we have assumed that membership in the relation *is* decidable). Thus, the minimal environments consistent with a relation for propositional variables and formulas are defined as follows.

Given variables x, a and a relation R relating variables bound in two lists of Boolean equations e_1, e_2, we will in the lemmas of this part prove that it is decidable if a is reachable from x through R. Using this lemma, we can define the environment which is consistent with R and assigns *true* to the smallest number of variables as well as x. This environment is the environment assigning *true* to all variables reachable from x through R. However, since the lemma can only be used for relations relating bound variables, we need an additional parameter. This parameter is any object H of type *relates_bound* R e_1 e_2, i.e. H proves that R relates only variables bound in e_1 to variables bound in e_2. In this text we will write $\theta_{R,x}^{e_1,e_2,H}$ for *minimal_environment_for_var*(R, e_1, e_2, H, x). Note that if R only relates bound variables then $\theta_{R,x} = \theta_{R,x}^{e_1,e_2,H}$.

```
Definition minimal_environment_for_var
    (R:propVar_relation)(e1 e2:genBES)
        (H:relates_bound R e1 e2)(x:propVar) : environment:=
fix env (a:propVar) := if reachable_dec_b e1 e2 R H x a
                        then true
                        else beq_nat x a
```

Lemma 2.2.17 shows that, if some propositional formula g evaluates to *true* under the minimal environment of some clause f in some relation R, then g is a relative consequence of f. Thus, it would have been sufficient if this environment was only correctly defined on clauses. In Coq, we did not add the minimal environments variables occurring within the scope of a disjunction from the minimal environment consistent with a propositional formula. However, the properties of the definition are equivalent for clauses, and for clauses the definitions coincide.

We can lift the notion of a minimal environment for a variable under some relation to the minimal environment for a clause under some relation. This environment is in essence the union of the minimal environments of all variables used in the clause, and it is the minimal environment consistent with R such that the clause evaluates to *true*. In this text we will write $\theta_{R,f}^{e_1,e_2,H}$ for *minimal_environment_for_conj*(R, e_1, e_2, H, f). Note that, if f is a clause and R only relates bound variables, then $\theta_{R,f}^{e_1,e_2,H} = \theta_{R,f}$.

```
Fixpoint minimal_environment_for_conj
    (R:propVar_relation)(e1 e2:genBES)(H:relates_bound R e1 e2)
        (f:propForm)(x:propVar):bool :=
match f with
| var y => (minimal_environment_for_var R e1 e2 H y x)
| f1 /\p f2 => (minimal_environment_for_conj R e1 e2 H f1 x)
              ||(minimal_environment_for_conj R e1 e2 H f2 x)
| _ => false
end
```

Lemmas

Lemma reachable_trans

The relation induced by reachable is transitive, i.e. given a relation R and propositional variables x, y, z such that z is reachable from y through R, and y is reachable from x through R. Then z is also reachable from x through R.

$$\forall R, \forall\ x\ y\ z, R^+\ x\ y \rightarrow R^+\ y\ z \rightarrow R^+\ x\ z$$

Lemma reachable_maintains_relates_bound
Given a relation R which relates only bound variables from some list of Boolean equations e_1 to bound variables from another list of Boolean equations e_2, then the relation obtained from *reachable* on R also relates only bound variables from e_1 to bound variables from e_2.

$$\forall e_1 e_2, \forall R, R \subseteq e_1 \times e_2 \to R^+ \subseteq e_1 \times e_2$$

Lemma reachable_dec_b
Given a relation R relating only bound variables from some list of Boolean equations e_1 to bound variables from another list of Boolean equations e_2, it is decidable for any pair of propositional variables x, y whether or not y is reachable from x through R.

$$\begin{aligned}\forall e_1 e_2, \forall R, & R \subseteq e_1 \times e_2 \to \\ & \forall x\ y, \{R^+\ x\ y\} + \{\neg(R^+\ x\ y)\}\end{aligned}$$

Note: Here, the notation $\{P\} + \{Q\}$ means that we always either have P or we have Q.

Lemma minimal_environment_for_var_consistent
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional variables x, the minimal environment for x under R is consistent with R.

$$\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2, \forall x, \theta_{R,x}^{e_1,e_2,H} \in \Theta_R$$

Lemma minimal_environment_for_var_self
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional variables x, the minimal environment for x under R assigns *true* to x.

$$\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2, \forall x, \theta_{R,x}^{e_1,e_2,H}(x)$$

Lemma min_env_rel_cons
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional variables x, y such that y is a relative consequence of x under R, the minimal environment for x under R assigns *true* to y.

$$\forall e_1 e_2, \forall R, \forall H : R \subseteq e_1 \times e_2, \forall x\ y, (var\ x) \overset{R}{\Rightarrow} (var\ y) \to \theta_{R,x}^{e_1,e_2,H}(y)$$

Lemma reachable_if_min_env
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional variables x, y, if y is assigned *true* under the minimal environment assigning *true* to x under R, y is reachable through R from x.

$$\forall e_1 e_2, \forall R, \forall H : R \subseteq e_1 \times e_2, \forall x\ y, x \neq y \to \theta_{R,x}^{e_1,e_2,H}(y) \to R^+\ x\ y$$

Lemma rel_cons_min_env
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional variables x, y, if y is assigned *true* under the minimal environment assigning *true* to x under R, then y is a relative consequence of x under R.

$$\forall e_1 e_2, \forall R, \forall H : R \subseteq e_1 \times e_2, \forall x\ y, \theta_{R,x}^{e_1,e_2,H}(y) \rightarrow (var\ x) \overset{R}{\Rightarrow} (var\ y)$$

Lemma minimal_environment_for_conj_consistent
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional formulas f, the consistent environment of f in R, is consistent with R.

$$\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2, \forall f, \theta_{R,f}^{e_1,e_2,H} \in \Theta_R$$

Lemma minimal_environment_for_conj_self
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all clauses f not equivalent to $\bot$, the minimal environment consistent with R assigning *true* to all variables in f assigns *true* to f.

$$\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2, \forall f, clause(f) \rightarrow \neg(f \Leftrightarrow \bot) \rightarrow [\![f]\!]\theta_{R,f}^{e_1,e_2,H}$$

Lemma min_env_var_rel_cons_form
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all variables x and propositional formulas f, if f is *true* under the minimal environment consistent with R assigning *true* to x, then f is a consequence of x relative to R.

$$\forall e_1 e_2, \forall R, \forall H : R \subseteq e_1 \times e_2, \forall x, \forall f, [\![f]\!]\theta_{R,x}^{e_1,e_2,H} \rightarrow (var\ x) \overset{R}{\Rightarrow} f$$

Lemma var_rel_cons_form_min_env
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all variables x and propositional formulas f, if f is a consequence of x relative to R, then f is *true* under the minimal environment consistent with R assigning *true* to x.

$$\forall e_1 e_2, \forall R, \forall H : R \subseteq e_1 \times e_2, \forall x, \forall f, (var\ x) \overset{R}{\Rightarrow} f \rightarrow [\![f]\!]\theta_{R,x}^{e_1,e_2,H}$$

Lemma min_for_conj_rel_cons
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all variables x and propositional formulas f, if f is a consequence of x relative to R, then f is *true* under the minimal environment consistent with R assigning *true* to x.

$$\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2, \forall f g, clause(f) \Rightarrow [\![g]\!]\theta_{R,f}^{e_1,e_2,H} \rightarrow f \overset{R}{\Rightarrow} g$$

6.5.2 Induction on relations

In the previous section we introduced *relates_bound* as a proposition on a relation and two lists of Boolean equations, and introduced the assumption that if a relation satisfies this proposition for any pair of lists of Boolean equations, then membership in the relation for any pair of variables is decidable. This makes it possible for us to inductively define *relates_bound*. This is because, given two lists of Boolean equations l_1, l_2, the empty relation relates only variables bound in l_1 to variables bound in l_2. Furthermore, if a relation R relates only bound variables bound in l_1 to variables bound in l_2 then adding any pair of variables from $l_1 \times l_2$ which does not occur in R to R results in a relation relating only pairs of variables from $l_1 \times l_2$. We define this inductively as a relation R being a subset of $l_1 \times l_2$, i.e. *is_subset* $R\ l_1\ l_2$.

Normally, given two relations which are equivalent, we would treat these relations to be equal. However, this is not the case in Coq. For example, a relation R which does not relate some variables x to some variable y is not equal to $(R \cup \{x, y\}) \setminus \{x, y\}$. This is related to the fact that equality is not a very simple concept, consider $2 + 2$ and $1 + 3$, if we consider the semantics of these statements, then the statements are equal. However, if we consider the syntax of the statements, then the statements are clearly different. For more on equality and type theory, we refer the interested reader to [15].

What is important here is that we want the is_subset property not to distinguish between equivalent relations (i.e. relations which relate the same variables to the same variables). To fix this there are two options: either to add the assumption that two equivalent relations are equal, or by adding an additional constructor to *is_subset*. To minimize the number of assumptions, we chose to add an additional constructor stating that, if two relations are equivalent and one of them is a subset relation, then the other is also a subset relation (of the same two lists of Boolean equations). We will write $l_1 \times l_2 \supseteq R$ instead of *is_subset* $l_1\ l_2\ R$.

```
Inductive is_subset (l1 l2:list booleanEquation)
                                    : propVar_relation -> Prop :=
| is_empty : forall R:propVar_relation,
  (forall x y:nat, ~R x y) -> is_subset l1 l2 R
| not_empty : forall R:propVar_relation, is_subset l1 l2 R ->
  forall x y:nat, ~R x y -> bnd l1 x -> bnd l2 y ->
    is_subset l1 l2 (rel_union R (pair_relation x y))
| eqv : forall R1 R2:propVar_relation, rel_eqv R1 R2 ->
  is_subset l1 l2 R1 -> is_subset l1 l2 R2
```

In stead of building a relation in the natural way like in *is_subset*, where we build a relation by starting with the empty relation and adding elements until we reach the desired relation, we can also do this inversely (for relations $\subseteq l_1 \times l_2$). This can be done as follows. The maximal relation on two lists of Boolean equations is a subset of these two lists of Boolean equations. Furthermore, if a relation is a subset of two lists of Boolean equations, then the result of removing any pair of variables related by this relation is also a subset of these two lists. We define this manner of inductively building a subset relation for two lists of Boolean equations as *rev_subset*. Just like with *is_subset*, we also add the constructor stating that for any two equivalent relations, if one is a reverse subset on two lists of Boolean equations, then the other is also a reverse subset on these lists. We will write $l_1 \times l_2 \overset{rev}{\supseteq} R$ instead of *rev_subset* l_1 l_2 R.

```
Inductive rev_subset (l1 l2:list booleanEquation)
                                    : propVar_relation -> Prop :=
| is_full : forall R:propVar_relation,
  (forall x y:nat, R x y <-> (bnd l1 x /\ bnd l2 y)) ->
    rev_subset l1 l2 R
| not_full : forall R:propVar_relation, rev_subset l1 l2 R ->
  forall x y:nat, R x y -> rev_subset l1 l2 (remPoint R x y)
| eqv_rev : forall R1 R2:propVar_relation, rel_eqv R1 R2 ->
  rev_subset l1 l2 R1 -> rev_subset l1 l2 R2
```

The notations introduced in this section might cause some confusion for the reader since we use $\subseteq$, $\supseteq$ and $\overset{rev}{\supseteq}$ for representing different properties; $R \subseteq l_1 \times l_2$ stands for *relates_bound*$(R\ l_1\ l_2)$, $l_1 \times l_2 \supseteq R$ stands for *is_subset*$(l_1\ l_2\ R)$ and $l_1 \times l_2 \overset{rev}{\supseteq} R$ stands for *rev_subset* $(l_1\ l_2\ R)$. However, these definitions are all equivalent in this setting (as shown in the lemmas of this section), so the confusion should not be (too) much of a problem.

Finally, we can define the inverse size of a relation as the cardinality of a reverse subset, as in Definition 5.3.2. To avoid some issues with subtracting on natural numbers and the number 0, since $0-1$ is defined as 0 in Coq, we start counting the cardinality with 1. Thus, the size of the largest subset relation of two lists of Boolean equations (the maximal relation on these lists) is 1. Given a subset relation R of size n, the size of the result of removing any pair of variables related in R from R is $S(n)$ (the successor of n). Again, similar to *rev_subset* and *is_subset*, we add that, given two equivalent reverse subset relations, if the size of one relation is n, then the size of the other relation is also n. We will write $\#(l_1 \times l_2 \setminus R) = n$ for *rev_subset_card* l_1 l_2 R n. If no confusion is possible then we may write $\#R = n$.

```
Inductive rev_subset_card (l1 l2:list booleanEquation)
                              : propVar_relation -> nat -> Prop :=
| card_full : forall R:propVar_relation,
  (forall x y:nat, R x y <-> (bnd l1 x /\ bnd l2 y)) ->
    rev_subset_card l1 l2 R 1
| card_not_full : forall R:propVar_relation, forall n:nat,
  rev_subset_card l1 l2 R n -> forall x y:nat, R x y ->
    rev_subset_card l1 l2 (remPoint R x y) (S n)
| card_eqv : forall R1 R2:propVar_relation, forall n:nat,
  rev_subset_card l1 l2 R1 n -> rel_eqv R1 R2 ->
    rev_subset_card l1 l2 R2 n
```

This seems like a very roundabout way of specifying *rev_subset_card*. We first defined being a subset, then being a reverse subset, to finally define the cardinality of reverse subsets. However, adding the definitions *is_subset* and *rev_subset* greatly simplifies the proof of $R \subseteq l_1 \times l_2$ being equivalent to there being some n such that $\#(l_1 \times l_2 \setminus R) = n$, since it allows us to tackle proving this via simpler steps; first proving that *relates_bound* and *is_subset* are equivalent notions, then showing that *is_subset* and *ref_subset* are equivalent, and finally showing that for all relations which are a *rev_subset* there is some n which is the cardinality of this relation.

The inspiration for the definitions from this section came from the Finite set library in Coq[16], which defines finite sets and their cardinality in a similar fashion.

Lemmas

Lemma bound_then_is_subset
For any relation R and lists of Boolean equations l_1, l_2, if R only relates variables bound in l_1 to variables bound in l_2, then *is_subset* l_1 l_2 R.

$$\forall l_1 l_2, \forall R, (\forall xy, R\ x\ y \rightarrow (x \in l_1 \wedge y \in l_2)) \rightarrow l_1 \times l_2 \supseteq R$$

Lemma max_rel_inverse
For any relation R and lists of Boolean equations l_1, l_2, if *is_subset* l_1 l_2 R then the maximal relation between l_1 and l_2 minus R is a reverse subset relation, i.e. *rev_subset* l_1 l_2 $((l_1 \times l_2) \setminus R)$.

$$\forall l_1 l_2, \forall R, l_1 \times l_2 \supseteq R \rightarrow l_1 \times l_2 \overset{rev}{\supseteq} ((l_1 \times l_2) \setminus R)$$

Lemma rev_subset_then_bound
For any relation R and lists of Boolean equations l_1, l_2, if *rev_subset* l_1 l_2 R then R only relates variables bound in l_1 to variables bound in l_2.

$$\forall l_1 l_2, \forall R, l_1 \times l_2 \overset{rev}{\supseteq} R \rightarrow \forall x\ y, R\ x\ y \rightarrow (x \in l_1 \wedge y \in l_2)$$

Lemma rev_subset_rev_subset_card
For any relation R and lists of Boolean equations l_1, l_2, if *rev_subset* l_1 l_2 R then there exists some n such that *rev_subset_card* l_1 l_2 R n.

$$\forall l_1 l_2, \forall R, l_1 \times l_2 \overset{rev}{\supseteq} R \rightarrow \exists n, \#(l_1 \times l_2 \setminus R) = n$$

Lemma rev_subset_card_rev_subset
For any relation R, lists of Boolean equations l_1, l_2 and natural number n, if *rev_subset_card* l_1 l_2 R n, then R is a reverse subset on $l_1 \times l_2$.

$$\forall l_1 l_2, \forall n, \forall R, \#(l_1 \times l_2 \setminus R) = n \rightarrow l_1 \times l_2 \overset{rev}{\supseteq} R$$

Lemma rev_subset_decreasing
Take any relation Γ', lists of Boolean equations l_1, l_2 and natural number n such that *rev_subset_card* l_1 l_2 Γ' n. Then, for any relation Γ and variables x, y bound in l_1 resp. l_2 such that Γ is equivalent to the union of Γ' with $\{x, y\}$, and Γ relates x to y, but Γ' does not, we have that the reverse size of Γ is one smaller than the reverse size of Γ', i.e. *rev_subset_card* l_1 l_2 Γ $(n-1)$.

$$\begin{aligned} \forall n, \forall l_1 l_2, \forall \Gamma', \#(l_1 \times l_2 \setminus \Gamma') = n \rightarrow \\ \forall \Gamma, \forall x\ y, \neg(\Gamma'\ x\ y) \rightarrow \Gamma\ x\ y \rightarrow x \in l_1 \rightarrow y \in l_2 \rightarrow \\ \Gamma \sim (\Gamma' \cup \{x, y\}) \rightarrow \#(l_1 \times l_2 \setminus \Gamma) = n - 1 \end{aligned}$$

6.5.3 Relative consequence decidability

The subset and reverse subset properties defined in the previous section have been defined on lists of Boolean equations. To obtain their definitions on *genBES*, we define a function *flatten_genBES* which obtains the list of all Boolean equations in the blocks from a list of blocks.

```
Fixpoint flatten_genBES (e1:list block):list booleanEquation :=
match e1 with
| nil => nil
| bl::e' => bl ++ (flatten_genBES e')
end.
```

Lemma rel_cons_decidable
For any relation R relating only pairs of variables from $e_1 \times e_2$ and for any propositional formulas f, g, either g is a relative consequence of f under R or not.

$$\forall R, \forall e_1 e_2, R \subseteq (\mathit{flatten\_genBES}\ e_1) \times (\mathit{flatten\_genBES}\ e_2) \Rightarrow \\ \forall f g, (f \overset{R}{\Rightarrow} g \vee \neg(f \overset{R}{\Rightarrow} g))$$

6.6 Well-formed BES

Given a list of blocks, we define the property that all variables bound in this list only occur on the left hand side of equations once as *w_d*.

```
Definition w_d(E:list block) :=
                      forall x:propVar, bnd E x <-> bnd_cnt E x=1
```

Using this definition, a well-formed BES (or *BES*) is a tuple of a *genBES* and a proof that this *genBES* is well-formed.

```
Record BES : Set := makeBES {
  bes:> genBES;
  well_defined: w_d bes}
```

Given a list of Boolean equations, we define a recursive function *rhs_block* which finds the right-hand side of the first equation, where the left-hand side is some specified variable. If the variable is not bound in the list, then the proposition consisting of the specified variable is returned.

```
Fixpoint rhs_block (l:list booleanEquation)(x:propVar)
                                                      : propForm :=
match l with
| nil => var x
| cons (bEqn xc f) l' => if beq_nat xc x
                         then f
                         else rhs_block l' x
end
```

We lift *rhs_block* from lists of Boolean equations to BES in the recursive function *rhs*.

```
Fixpoint rhs (E:BES)(x:propVar) : propForm :=
let fix helper (l:list block) : propForm :=
match l with
| nil => bot
| bl::l' => if (bnd_block_dec bl x)
            then rhs_block bl x
            else helper l'
end in helper E
```

We define a function *bnd_block_number* on a list of blocks and a propositional variable, to return the length of the list plus 1 if the variable is not bound in the list, and otherwise to return the lowest index of a block in which the variable is bound.

```
Fixpoint bnd_block_number (e:list block)(x:propVar):=
match e with
| nil => 0
| bl::e' => if (bnd_block_dec bl x)
            then 0
            else S(bnd_block_number e' x)
end
```

Using the previous function, defining a function which determines the *rank* of a variable in a BES is simple. This function corresponds with the Definition from 2.3.5. Note that the standard definition of rank starts counting at 0 resp. 1 if we start with a greatest resp. least fixed point sign. However, we reserve 0 for unbound variables. Thus, to maintain that greatest fixed points correspond with even rank, we start counting at 2 if the first block has a greatest fixed point, and 1 otherwise.

```
Definition rank (E:list block)(x:propVar) : nat :=
if (bnd_dec E x)
then match E with
     | nil => 0
     | bl::_ => if blockType bl
                then S(S(bnd_block_number E x))
                else S(bnd_block_number E x)
     end
else 0
```

Lemmas

Lemma 6.6.1. eq_rank
For any list of blocks e and propositional variables x, y of equal rank, one of them is bound in e if and only if the other is as well.

$$\forall e, \forall x\ y, rank\ e\ x = rank\ e\ y \to (x \in e \leftrightarrow y \in e)$$

6.6.1 Consistent consequences

Given a BES E, a relation on propositional variables R and propositional variables x, y, if for any environment consistent with R, if the solution of the right-hand side of x in E under this environment implies the solution of the right-hand side of y in E under this environment then we say *consistent_environment_rhs* $E\ R\ x\ y$.

```
Definition consistent_environment_rhs (E:BES)(R:propVar_relation)
                                                  (x y:propVar) :=
forall theta:environment, consistent_environment theta R ->
  [[rhs E x]]theta -> [[rhs E y]]theta
```

Given a BES E and a relation on propositional variables R, if all variables related by R have the same rank, and for pairs of bound variables the right-hand sides of these variables in E are relative consequences under R, then R is a consistent consequence relation. We may write *cc* $E\ R$ to indicate that R is a consistent consequence on E. *consistent_consequence* corresponds with the Definition from 3.1.1.

```
Definition consistent_consequence (E:BES)(R:propVar_relation)
                                                        : Prop :=
forall x y:propVar, R x y ->
  (rank E x=rank E y)
/\
  (bnd E x -> bnd E y -> consistent_environment_rhs E R x y)
```

Given a BES E, we say that propositional variables x, y are related in *cc_max* E if there exists some relation R which is a consistent consequence relation on E and relates only variables bound in E to variables bound in E, such that $(x, y) \in R$. The definition for $\lessdot^E$ corresponds with *cc_max* E.

```
Definition cc_max (E:BES)(x:propVar)(y:propVar) : Prop :=
exists R:propVar_relation, consistent_consequence E R /\
  relates_bound R (flatten_genBES E) (flatten_genBES E) /\ R x y
```

Lemmas

Lemma 6.6.2. cc_max_consistent_environment_has_cc
For any BES E and environment θ, if θ is consistent with $\lessdot^E$ then for any propositional variables x, y related by $\lessdot^E$ there exists a relation R which is a consistent consequence relation on E relating x to y such that θ is consistent with R.

$$\forall E, \forall \theta, \theta \in \Theta_{\lessdot^E} \Rightarrow \forall x\ y, x\ \lessdot^E\ y \rightarrow \exists R, (cc\ E\ R \wedge R\ x\ y \wedge \theta \in \Theta_R)$$

Lemma 6.6.3. cc_max_is_cc
For any BES E, $\lessdot^E$ is a consistent consequence relation on E.

$$\forall E, cc\ E\ (\lessdot^E)$$

6.6.2 Relative consistent consequences

By relaxing the second requirement of consistent consequence relations, which states that the right-hand sides of related bound variables should be relative consequences, to allow them to be relative consequences relative to the union with another relation, we will later be able to prove certain properties more naturally. The new definition is as follows.

```
Definition relative_cc (E:BES)(R G:propVar_relation) : Prop :=
forall x y:propVar, R x y ->
  (rank E x=rank E y)
 /\
  (bnd E x -> bnd E y ->
    consistent_environment_rhs E (R U G) x y)
```

If we restrict the relative consistent consequence relation to relations on bound variables, then we can define the largest consistent consequence relation relative to some other relation. This relation relates any pair of variables if they are either: related by the identity relation, related by the relative relation, or related by some relation which is a consistent consequence relation relative to the relative relation. For any BES E and relation Γ, *rel_cc_on_propVar* E Γ corresponds with the definition of $\lessdot^E_\Gamma$.

```
Definition rel_cc_on_propVar (E:BES)(G:propVar_relation)
                                            (x y:propVar) : Prop :=
exists R:propVar_relation,
  relates_bound R (flatten_genBES E) (flatten_genBES E) /\
  relative_cc E R G /\ (R U G U id_rel) x y
```

We can also define relative consistent consequence on propositional formulas. The following corresponds with the Definition in 5.0.4.

```
Definition rel_cc_on_propForm (E:BES)(G:propVar_relation)
                                        (f g:propForm) : Prop :=
exists R:propVar_relation,
  relates_bound R (flatten_genBES E) (flatten_genBES E) /\
  relative_cc E R G /\ (forall theta:environment,
    consistent_environment theta (R U G) ->
      [[f]]theta -> [[g]]theta)
```

Lemmas

Lemma 6.6.4. empty_relation_rel_cc
The empty relation is a consistent consequence relation on any BES, relative to any other relation Γ.

$$\forall E, \forall \Gamma, \mathit{relative\_cc}\ E\ \emptyset\ \Gamma$$

Lemma 6.6.5. empty_relative_cc
If a relation R is a consistent consequence relation on some BES E relative to the empty relation, then R is a consistent consequence relation on E.

$$\forall E, \forall R, \mathit{relative\_cc}\ E\ R\ \emptyset \rightarrow \mathit{cc}\ E\ R$$

Lemma 6.6.6. cc_relative_empty
If a relation R is a consistent consequence relation on some BES E, then R is a consistent consequence relation on E relative to the empty relation.

$$\forall E, \forall R, \mathit{cc}\ E\ R \rightarrow \mathit{relative\_cc}\ E\ R\ \emptyset$$

Lemma 6.6.7. propForm_cons_cc_max
Given a BES E and variables x, y bound in E, then if propositional formula y is a consistent consequence of propositional formula x relative to the empty relation, then y is a consistent consequence of x.

$$\forall E, \forall x\ y, x \in E \rightarrow y \in E \rightarrow (\mathit{var}\ x) \overset{\lessdot_{\emptyset}^{E}}{\Longrightarrow} (\mathit{var}\ y) \rightarrow x\ \lessdot^{E}\ y$$

6.7 Proof system for consistent consequences

We define a statement to be a tuple of a relation on propositional variables and two propositional formulas.

```
Inductive statement :=
| stmt : propVar_relation -> propForm -> propForm -> statement.
```

We introduce some notation in Coq for *stmt* Γ f g as follows. However, we will write $\Gamma \vdash f \subset g$ for *stmt* Γ f g in this text.

```
Notation "G |- f --> g" := (stmt G f g)(at level 50).
```

A proof tree can be built by applying rules until we close all branches. We can inductively define proof trees, by specifying a constructor for each of the rules in the proof system. The inspiration for the type of the proof system came from the proof system for CTL presented in [3].

As an example, consider the following constructor.

```
| TRA (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a b) -> prv_tree E (stmt G b c) ->
    prv_tree E (stmt G a c))
```

This constructor corresponds with the transitivity rule in the proof system. It states that we can build a proof tree *prv_tree* E $G \vdash a \subset c$ (i.e. a proof tree with $G \vdash a \subset c$ as the conclusion), by supplying constructor TRA with proof trees *prv_tree* E $G \vdash a \subset b$ and *prv_tree* E $G \vdash b \subset c$.

```
Inductive prv_tree (E:BES) : statement -> Prop :=
| AS1 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp a (andp b c)) (andp (andp a b) c)))
| AS2 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp (andp a b) c) (andp a (andp b c))))
| AS3 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a (orp b c)) (orp (orp a b) c)))
| AS4 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp (orp a b) c) (orp a (orp b c))))

| COM1 (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp a b) (andp b a)))
| COM2 (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a b) (orp b a)))

| DS1 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a (andp b c))
                                  (andp (orp a b) (orp a c))))
| DS2 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp (orp a b) (orp a c))
                                          (orp a (andp b c))))
| DS3 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp a (orp b c))
                                 (orp (andp a b) (andp a c))))
| DS4 (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp (andp a b) (andp a c))
                                              (andp a (orp b c))))

| AB1 (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a (andp a b)) a))
| AB2 (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a (orp a (andp a b))))

| ID1 (a:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a (andp a a)))
| ID2 (a:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a a) a))

| SUP (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a (orp a b)))
| INF (a b:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (andp a b) a))

| TOP (a:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a (andp a top)))
| BOT (a:propForm)(G:propVar_relation):
  (prv_tree E (stmt G (orp a bot) a))

| CTX (a b c:propForm)(x:propVar)(G:propVar_relation):
  (prv_tree E (stmt G a b) ->
    prv_tree E (stmt G (replace c x a) (replace c x b)))
| TRA (a b c:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a b) -> prv_tree E (stmt G b c) ->
    prv_tree E (stmt G a c))
| REF (a:propForm)(G:propVar_relation):
  (prv_tree E (stmt G a a))

| CC (x y:propVar)(G:propVar_relation):
  (((bnd (bes E) x) /\ (bnd (bes E) y) /\ (rank E x=rank E y)) ->
    (prv_tree E (stmt (rel_union G (pair_relation x y))
                                        (rhs E x)(rhs E y))) ->
      (prv_tree E (stmt G (var x) (var y))))
| CNT (x y:propVar)(G:propVar_relation):
  ((G x y) -> (prv_tree E (stmt G (var x) (var y))))
.
```

6.7.1 Soundness

For proving the soundness of the proof system, we define a converter which takes a BES E and a statement '*stmt G f g*', and returns *rel_cc_on_propForm E G f g*. This ensures that Coq applies the induction principle correctly, when we apply induction on the structure of proof trees to prove soundness of the proof system.

```
Definition mk_rel_cc(E:BES)(s : statement) : Prop :=
match s with
| stmt G f g => rel_cc_on_propForm E G f g
end.
```

Lemmas

Lemma 6.7.1. soundness
For any BES E, relation on propositional variables G and propositional formulas f, g, if we have a proof tree with $G \vdash f \subset g$ as the root, then g is a consistent consequence of f on E, relative to G.

$$\forall E, \forall f g, \forall G, \mathit{prv\_tree}\ E\ (G \vdash f \subset g) \to \mathit{mk\_rel\_cc}\ E\ (G \vdash f \subset g)$$

Lemma 6.7.2. prv_system_sound
For any BES E, relation on propositional variables G and propositional variables x, y bound in E, if we have a proof tree with $\emptyset \vdash x \subset y$ as the root, then y is a consistent consequence of x on E.

$$\forall E, \forall x\ y, x \in E \to y \in E \to \mathit{prv\_tree}\ E\ (\emptyset \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y)) \to x\ \lessdot^{E}\ y$$

6.7.2 Complete for logical consequence

Lemmas

Lemma 6.7.3. cSPLIT
For any BES E, relation on propositional variables G and propositional formulas f, g_1, g_2, if we can build proof trees with $G \vdash f \subset g_1$ and $G \vdash f \subset g_2$ as the root, then we can build a proof tree with $G \vdash f \subset g_1 \wedge g_2$ as the root.

$$\forall E, \forall G, \forall f\ g_1\ g_2, \mathit{prv\_tree}\ E\ (G \vdash f \subset g_1) \to \mathit{prv\_tree}\ E\ (G \vdash f \subset g_2) \to \\ \mathit{prv\_tree}\ E\ (G \vdash f \subset (g_1 \wedge_p g_2))$$

Lemma 6.7.4. cGROW_L
For any BES E, relation on propositional variables G and propositional formulas f, g_1, g_2, if we can build a proof tree with $G \vdash f \subset g_1$ as the root, then we can build a proof tree with $G \vdash f \subset g_1 \vee g_2$ as the root.

$$\forall E, \forall G, \forall f\ g_1\ g_2, \mathit{prv\_tree}\ E\ (G \vdash f \subset g_1) \to \mathit{prv\_tree}\ E\ (G \vdash f \subset (g_1 \vee_p g_2))$$

Lemma 6.7.5. cGROW_R
For any BES E, relation on propositional variables G and propositional formulas f, g_1, g_2, if we can build a proof tree with $G \vdash f \subset g_2$ as the root, then we can build a proof tree with $G \vdash f \subset g_1 \vee g_2$ as the root.

$$\forall E, \forall G, \forall f\ g_1\ g_2, \mathit{prv\_tree}\ E\ (G \vdash f \subset g_2) \to \mathit{prv\_tree}\ E\ (G \vdash f \subset (g_1 \vee_p g_2))$$

Lemma 6.7.6. aSPLIT
For any BES E, relation on propositional variables G and propositional formulas f_1, f_2, g, if we can build proof trees with $G \vdash f_1 \subset g$ and $G \vdash f_2 \subset g$ as the root, then we can build a proof tree with $G \vdash f_1 \vee f_2 \subset g$ as the root.

$$\forall E, \forall G, \forall f_1\ f_2\ g, \mathit{prv\_tree}\ E\ (G \vdash f_1 \subset g) \rightarrow \mathit{prv\_tree}\ E\ (G \vdash f_2 \subset g) \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (f_1 \vee_p f_2) \subset g)$$

Lemma 6.7.7. aGROW_L
For any BES E, relation on propositional variables G and propositional formulas f_1, f_2, g, if we can build a proof tree with $G \vdash f_1 \subset g$ as the root, then we can build a proof tree with $G \vdash f_1 \wedge f_2 \subset g$ as the root.

$$\forall E, \forall G, \forall f_1\ f_2\ g, \mathit{prv\_tree}\ E\ (G \vdash f_1 \subset g) \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (f_1 \wedge_p f_2) \subset g)$$

Lemma 6.7.8. aGROW_R
For any BES E, relation on propositional variables G and propositional formulas f_1, f_2, g, if we can build a proof tree with $G \vdash f_2 \subset g$ as the root, then we can build a proof tree with $G \vdash f_1 \wedge f_2 \subset g$ as the root.

$$\forall E, \forall G, \forall f_1\ f_2\ g, \mathit{prv\_tree}\ E\ (G \vdash f_2 \subset g) \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (f_1 \wedge_p f_2) \subset g)$$

Lemma 6.7.9. distribute_prv_tree
For any BES E, relation on propositional variables G and DNF's f, g, we can build a proof tree with $G \vdash f \wedge g \subset (\mathit{dist}(f)(g))$ as the root.

$$\forall E, \forall G, \forall g, \mathit{DNF}(g) \rightarrow \forall f, \mathit{DNF}(f) \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (f \wedge_p g) \subset \mathit{dist}(f)(g))$$

Lemma 6.7.10. complete_propForm_cons_DNF
For any BES E, relation on propositional variables G and DNF f, we can build a proof tree with $G \vdash f \subset \mathit{toDNF} f$ as the root.

$$\forall E, \forall G, \forall f, \mathit{prv\_tree}\ E\ (G \vdash f \subset \mathit{toDNF}(f))$$

Lemma 6.7.11. complete_DNF_cons_propForm
For any BES E, relation on propositional variables G, DNF f and propositional formula g, if g is a logical consequence of f, then we can build a proof tree with $G \vdash f \subset g$ as the root.

$$\forall E, \forall G, \forall g\ f, \mathit{DNF}(f) \rightarrow f \Rightarrow g \rightarrow \mathit{prv\_tree}\ E\ (G \vdash f \subset g)$$

Lemma 6.7.12. complete_cons
For any BES E, relation on propositional variables G and propositional formulas f, g, if g is a logical consequence of f, then we can build a proof tree with $G \vdash f \subset g$ as the root.

$$\forall E, \forall G, \forall g\ f, f \Rightarrow g \rightarrow \mathit{prv\_tree}\ E\ (G \vdash f \subset g)$$

6.7.3 Complete for consistent consequence

Lemmas

Lemma 6.7.13. lem5
For any BES E, relation R on variables bound in E, relation G on propositional variables and propositional formulas f, g such that g is a consequence of f relative to R, if we can make proof trees for all variables $(x, y) \in R$ with root $G \vdash x \subset y$, then we can create a proof tree with $G \vdash f \subset g$ as the root.

$$\begin{aligned}\forall E, \forall R\ G, R \subseteq (\mathit{flatten\_genBES}\ E) \times (\mathit{flatten\_genBES}\ E) \rightarrow \\ \forall f\ g, f \overset{R}{\Rightarrow} g \rightarrow (\forall x\ y, R\ x\ y \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y))) \rightarrow \\ \mathit{prv\_tree}\ E\ (G \vdash f \subset g)\end{aligned}$$

Lemma 6.7.14. complete_propVar
For any n, BES E, relation G on variables bound in E such that the size of the maximal relation minus G is n and propositional variables x, y such that $x \lessdot_G^E y$ we can construct a proof tree with $G \vdash x \subset y$ as the root.

$$\begin{aligned}\forall E, \forall n, \forall G, \#((\mathit{flatten\_genBES}\ E) \times (\mathit{flatten\_genBES}\ E) \setminus G) = n \rightarrow \\ \forall x\ y, x\ \lessdot_G^E\ y \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y))\end{aligned}$$

Lemma 6.7.15. prv_system_complete
For BES E, and propositional variables x, y such that $x\ \lessdot^E\ y$ we can construct a proof tree with $\emptyset \vdash x \subset y$ as the root.

$$\forall E, \forall x\ y, x\ \lessdot^E\ y \rightarrow \mathit{prv\_tree}\ E\ (\emptyset \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y))$$

6.7.4 Complete and sound

Lemmas

Lemma 6.7.16. prv_system_sound_and_complete
For any BES E and propositional variables x, y bound in E, y is a consistent consequence of x in E if and only if we can build a proof tree with $\emptyset \vdash x \subset y$ as the root.

$$\forall E, \forall x\ y, x \in E \rightarrow y \in E \rightarrow (x\ \lessdot^E\ y \leftrightarrow \mathit{prv\_tree}\ E\ (\emptyset \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y)))$$

6.8 Semantics of BES

We originally defined the semantics of a propositional formula under some environment to be in *Prop*. However, the semantics of a propositional formula is, in fact, a Boolean value. We can define a recursive function *b_propForm_solution* equivalent to *propForm_solution* on a propositional formula and an environment, obtaining a Boolean representing the solution of the propositional formula under the environment. In this text, we will indicate *b_propForm_solution* f θ with $[\![f]\!]_b\theta$

```
Fixpoint b_propForm_solution(f:propForm)(theta:environment)
                                                           : bool :=
match f with
| top => true
| bot => false
| var x => theta x
| f1 /\p f2 =>
     b_propForm_solution f1 theta && b_propForm_solution f2 theta
| f1 \/p f2 =>
     b_propForm_solution f1 theta || b_propForm_solution f2 theta
end.
```

We define a function for the unfolding of a list of Boolean equations in an environment. This function corresponds with Definition 2.3.11.

```
Fixpoint unfold_block (bl:list booleanEquation)
                              (theta:environment) : environment:=
match bl with
| nil => theta
| bEqn x fx::bl' =>
  redefineEnvironment (unfold_block bl' theta) x ([[fx]]_b theta)
end.
```

We define a function for redefining an environment θ (*result_unbound*) in the bound variables from some list of Boolean equations *bl* with their interpretations under another environment θ' (*result_bound*).

```
Definition redefine_bound (bl:list booleanEquation)
        (result_unbound result_bound:environment) : environment:=
fun (x:propVar) => if bnd_block_dec bl x
                   then result_bound x
                   else result_unbound x.
```

We define well-formedness of blocks as follows; all variables bound in the block occur exactly once at the left-hand side of an equation in the block.

```
Definition w_d_block(bl:list booleanEquation) :=
forall x:propVar, bnd_block bl x <-> bnd_cnt_block bl x=1.
```

We define a function *sol_iterator*, which applies a function on environments, H, i times to some environment *theta_b*. We write H^i(*theta_b*) instead of *sol_iterator* H *theta_b* i, as defined in Section 2.3.1.

```
Fixpoint sol_iterator (H:environment -> environment)
                                  (theta_b:environment)(i:nat) :=
match i with
| 0 => H theta_b
| S(k) => H (sol_iterator H theta_b k)
end.
```

We define a recursive function *genBES_solution* for obtaining the semantics of a list of blocks e under some environment θ.

```
Fixpoint genBES_solution (e:list block)(theta:environment)
                                      {struct e} : environment :=
let F (e'':list block)(B'':block)(theta'' theta_b'':environment)
                                                : environment :=
  unfold_block B'' (genBES_solution e''
                         (redefine_bound B'' theta'' theta_b''))
in
match e with
| nil => theta
| cons bl e' => if (blockType bl)
              then genBES_solution e' (redefine_bound bl theta
     (sol_iterator (F e' bl theta) full_environment (length bl)))
              else genBES_solution e' (redefine_bound bl theta
    (sol_iterator (F e' bl theta) empty_environment (length bl)))
end.
```

The semantics of a BES under some environment is simply the semantics given by *genBES_solution* on this BES and environment. The semantics given in Definition 2.3.12 corresponds with this definition.

```
Definition BES_solution (E:BES)(Theta:environment)
                                                : environment :=
  genBES_solution (the_genBES (bes E)) Theta.
```

Lemmas

Lemma 6.8.1. rank_bound_split
For any BES E, relation on propositional variables R which is a consistent consequence relation on E and variables $(x, y) \in R$, either both x and y are bound in E, or neither of them is.

$$\forall E, \forall R, cc\ E\ R \rightarrow \forall x\ y, R\ x\ y \rightarrow ((x \in E \wedge y \in E) \vee (x \notin E \wedge y \notin E))$$

Lemma 6.8.2. useful_unfold_block
For any BES consisting of a single block *bl*, any relation R on propositional variables which is a consistent consequence relation on this BES, and any environment θ consistent with R, the result of unfolding *bl* in θ is also consistent with R.

$$\begin{aligned}&\forall bl, \forall alt\_bl : alternating\_list\_block\ (bl :: nil),\\ &\forall w\_d\_bl : w\_d\ (make\_genBES\ (bl :: nil)\ alt\_bl),\\ &\forall R, cc\ (makeBES\ (make\_genBES\ (bl :: nil)\ alt\_bl)\ w\_d\_bl)\ R \rightarrow\\ &\forall \theta, \theta \in \Theta_R \rightarrow (||bl||\theta) \in \Theta_R\end{aligned}$$

Lemma 6.8.3. useful_sol_iterator
For any function f on environments, any relation R on propositional variables such that f maintains consistency of environments with R, any number n and any environment θ consistent with R, the result of applying f n times to θ is also consistent with R.

$$\begin{aligned}\forall f : \mathit{environment} \to \mathit{environment}, \forall R, (\forall \theta, \theta \in \Theta_R \to (f(\theta) \in \Theta_R)) \to \\ \forall n, \forall \theta, \theta \in \Theta_R \to (f^n(\theta)) \in \Theta_R\end{aligned}$$

Note, f : environment $\to$ environment should be read as f is a function mapping environments to environments, not f is a proof that environment implies environment.

Lemma 6.8.4. redef_bnd_consistent
For any BES consisting of a single block *bl*, relation R on propositional variables which is a consistent consequence relation on *bl*, and environments θ_u, θ_b consistent with R, the result of redefining θ_u in the variables bound in *bl* with the values of their interpretations under θ_b is also consistent with R.

$$\begin{aligned}\forall bl, \forall alt\_bl : \mathit{alternating\_list\_block}\ (bl :: nil), \\ \forall w\_d\_bl : w\_d\ (\mathit{make\_genBES}\ (bl :: nil)\ alt\_bl), \\ \forall R, cc\ (\mathit{makeBES}\ (\mathit{make\_genBES}\ (bl :: nil)\ alt\_bl)\ w\_d\_bl)\ R \to \\ \forall \theta_b\ \theta_u, \theta_b \in \Theta_R \to \theta_u \in \Theta_R \to \\ (\mathit{redefine\_bound}\ bl\ \theta_b\ \theta_u) \in \Theta_R\end{aligned}$$

Lemma 6.8.5. useful
For any BES E, relation R on propositional variables which is a consistent consequence relation on E and environment θ consistent with R, the solution of E in θ is also consistent with R.

$$\forall E, \forall R, cc\ E\ R \to \forall \theta, \theta \in \Theta_R \to ([\![E]\!]\theta) \in \Theta_R$$

Chapter 7

Conclusion

In [10], a proof system was presented for deriving consistent consequences between propositional variables based on syntax. However, to prove the proof system complete, an additional rule was added which turned out to be unsound. The goal of this project was to fix this issue, and define a proof system for deriving consistent consequences which is sound and complete. Furthermore, since in the past some errors had been made while trying to establish such a proof system, we decided to formalize the theory in Coq.

In this project, we have shown that the proof system from [10] can be made sound by removing the unsound rule (Chapter 4), and we have shown that the resulting proof system is complete for consistent consequence (Chapter 5). Furthermore, we have also shown that the proof system is complete for logical consequence, which was required for showing the completeness of the proof system (Chapter 5). Each of these proofs were formalized in Coq, and in this report we gave translations from the Coq formalization (documented in Chapter 6) to more abstract mathematical notation.

To paint the full picture, we also formalized the relation between the semantics of BES and consistent consequence relation (Chapter 3). To make the formalization in Coq simpler, we introduced an alternative semantics for BES, which we have shown to be equivalent to the standard semantics. This formalization revealed a small mistake in an earlier proof, and we have shown how to fix this issue by tweaking the definition of consistent consequence.

While working with Coq, we sometimes felt the desire to add additional axioms for results which seemed obvious (for example, the completeness of the proof system with regards to logical consequence). However, we have suppressed this desire to make sure that everything is formally verified.

Being forced to pay attention to all details has had both down- and upsides. It was difficult to define the notion of the inverse size of a relation which only relates from some finite set of variables, where we had to define *relates_bound*, *is_subset* and *rev_subset*. In Coq, showing that these notions are equivalent was not an easy task, while in Chapter 5 we did not even need to refer to these concepts since we could immediately define the inverse size of a relation. On the other hand, being forced not to ignore any of the details has made the intricate reasoning why the properties we have proven hold extremely clear. Furthermore, the way Coq manages the context is an asset during long complex proofs. During more complex proofs, Coq allows the user to easily break down the proof into smaller, bite sized parts (or, sub-goals). So, though Coq forces you to pay attention to details, it also assists the user with tackling complex proofs one step at a time, and makes sure you don't skip any of the details. In particular, this attention to detail has revealed small errors from previous work, such as the issue we found in the proof of the relation between the semantics of BES and the notion of consistent consequence.

7.1 The future: From BES to PBES

Consistent consequence was originally defined for simplifying soundness proofs of abstraction techniques for *PBES* and not *BES*. Therefore it is interesting to investigate if and how the proof system in this document can serve as a basis for a proof system in the setting of PBES. In Appendix C, we made an attempt at making this adjustment. The proof system in Appendix C is based on the one presented in this document, however the CC and CNT rules have been changed significantly to allow making derivations under some assumptions on data terms. Furthermore, we add rules to handle data terms, Boolean expressions, quantifiers, and manipulating the assumptions made on data terms, which we do not encounter in the setting of BES.

In Appendix C, we indicate the properties on which the CC and CNT rules in the setting of BES are based. We then show how we can formulate these properties in the setting of PBES in such a way that we can add assumptions on data terms to the system.

Next we discuss a set of rules for data terms and quantifiers, which are based on standard rules from sequent calculus [11]. Sequent calculus provides a method for deriving logical implications between predicate formulas, and part of the data rules in Appendix C were created by observing that in the proof system, what comes before the $\subset$ symbol is in essence a series of sequents (in $\Gamma \vdash f \subset g$, we can view both f and the contents of Γ as sequents).

This proof system appears to be a good basis for future work, and to show how it can be used we make an example derivation of consistent consequence on a PBES which cannot be instantiated into a BES. However, many details are glossed over or skipped altogether, and in particular the soundness of the proof system needs to be formally investigated.

Furthermore, if the system is indeed sound, it would be interesting to investigate how strong the system is in reality. Proving completeness of the proof system for BES was possible because in the setting of BES, $\lessdot$ is always finite. This is not the case for the setting of PBES, though this tactic could still be used if all consistent consequence relations can be finitely represented with the notation we introduced for the proof system: $\{X(x) \subset Y(y) \mid P\}$. However, it is likely that this is not the case, and it seems probable that no sound and complete proof system exists for deriving consistent consequences on PBES.

Appendix A

Full proof system

In this appendix we repeat the proof system for consistent consequences, together with all rules derived in lemmas and theorems for ease of reference while reading the soundness and completeness proofs.

Table A.1: Proof system for consistent consequences on BES $\mathcal{E}$; α, β and γ are arbitrary propositional formulas; X, Y are propositional variables; Γ is a an arbitrary relation on propositional variables; $\Gamma, X \subset Y$ is used to denote the relation $\Gamma \cup \{(X, Y)\}$

Axioms:
rules of the form $\overline{\Gamma \vdash A}$ *where A ranges over:*

AS1	$\alpha \wedge (\beta \wedge \gamma) \subset (\alpha \wedge \beta) \wedge \gamma$	DS1	$\alpha \vee (\beta \wedge \gamma) \subset (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
AS2	$(\alpha \wedge \beta) \wedge \gamma \subset \alpha \wedge (\beta \wedge \gamma)$	DS2	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma) \subset \alpha \vee (\beta \wedge \gamma)$
AS3	$\alpha \vee (\beta \vee \gamma) \subset (\alpha \vee \beta) \vee \gamma$	DS3	$\alpha \wedge (\beta \vee \gamma) \subset (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
AS4	$(\alpha \vee \beta) \vee \gamma \subset \alpha \vee (\beta \vee \gamma)$	DS4	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \subset \alpha \wedge (\beta \vee \gamma)$
COM1	$\alpha \wedge \beta \subset \beta \wedge \alpha$	AB1	$\alpha \vee (\alpha \wedge \beta) \subset \alpha$
COM2	$\alpha \vee \beta \subset \beta \vee \alpha$	AB2	$\alpha \subset \alpha \vee (\alpha \wedge \beta)$
ID1	$\alpha \subset \alpha \wedge \alpha$	ID2	$\alpha \vee \alpha \subset \alpha$
SUP	$\alpha \subset \alpha \vee \beta$	INF	$\alpha \wedge \beta \subset \alpha$
TOP	$\alpha \subset \alpha \wedge \top$	BOT	$\alpha \vee \bot \subset \alpha$

Logic rules:

TRA	$\dfrac{\Gamma \vdash \alpha \subset \beta \qquad \Gamma \vdash \beta \subset \gamma}{\Gamma \vdash \alpha \subset \gamma}$	REF	$\dfrac{}{\Gamma \vdash \alpha \subset \alpha}$
CTX	$\dfrac{\Gamma \vdash \alpha \subset \beta}{\Gamma \vdash \gamma[X := \alpha] \subset \gamma[X := \beta]}$		

Consistent Consequence rules:

CC $$\frac{\Gamma, X \subset Y \vdash f_X \subset f_Y}{\Gamma \vdash X \subset Y}\ \mathbf{rank}(X) = \mathbf{rank}(Y) \text{ and } X, Y \in \mathbf{bnd}_{\mathcal{E}}$$

CNT $$\frac{}{\Gamma \vdash X \subset Y}\ (X \subset Y \in \Gamma)$$

Split/grow rules:
In these rules, f, g, f_1, f_2, g_1, g_2 are propositional formulas

cSPLIT $$\frac{\Gamma \vdash f \subset g_1 \qquad \Gamma \vdash f \subset g_2}{\Gamma \vdash f \subset g_1 \wedge g_2}$$

cGROW$_L$ $$\frac{\Gamma \vdash f \subset g_1}{\Gamma \vdash f \subset g_1 \vee g_2}$$ cGROW$_R$ $$\frac{\Gamma \vdash f \subset g_2}{\Gamma \vdash f \subset g_1 \vee g_2}$$

aSPLIT $$\frac{\Gamma \vdash f_1 \subset g \qquad \Gamma \vdash f_2 \subset g}{\Gamma \vdash f_1 \vee f_2 \subset g}$$

aGROW$_L$ $$\frac{\Gamma \vdash f_1 \subset g}{\Gamma \vdash f_1 \wedge f_2 \subset g}$$ aGROW$_R$ $$\frac{\Gamma \vdash f_2 \subset g}{\Gamma \vdash f_1 \wedge f_2 \subset g}$$

DNF rules:
In these rules, f, g are propositional formulas

DIST $$\frac{}{\Gamma \vdash f \wedge g \subset \mathit{dist}(f)(g)}\ f, g \text{ are in DNF}$$

Lemma 5.2.10 $$\frac{}{\Gamma \vdash f \subset \mathit{DNF}(f)}$$

Lemma 5.2.11 $$\frac{}{\Gamma \vdash f \subset g}\ f \text{ is in DNF and } f \Rightarrow g$$

Lemma 5.2.12 $$\frac{}{\Gamma \vdash f \subset g}\ f \Rightarrow g$$

Completeness rules:
*In these rules, $R \subseteq$ **bnd** $\times$ **bnd** is a relation on propositional variables. Furthermore, f and g are propositional formulas.*

Lemma 5.3.1 $$\frac{\{\Gamma \vdash X \subset Y \mid (X, Y) \in R\}}{\Gamma \vdash f \subset g}\ f \xRightarrow{R \cup \Gamma} g$$

Lemma 5.3.3 $$\frac{}{\Gamma \vdash X \subset Y}\ X \lessdot_{\Gamma} Y$$

Appendix B

Equivalent semantics

In this appendix, we prove Lemma 2.3.13.

Lemma For any BES $\mathcal{E}$ and for any environment θ, $(\!|\mathcal{E}|\!)\theta = [\![\mathcal{E}]\!]\theta$.

Proof. Take some BES $\mathcal{E}$. We proceed by induction on the number of blocks in $\mathcal{E}$ (written $\#(\mathcal{E})$).

0 In this case, for any environment θ, $(\!|\varepsilon|\!)\theta = \theta = [\![\varepsilon]\!]\theta$

$k+1$ In this case, we have the following induction hypothesis: for any $\mathcal{E}'$ such that $\#(\mathcal{E}') = k$ and environment θ',

$$(\!|\mathcal{E}'|\!)\theta' = [\![\mathcal{E}']\!]\theta' \tag{1}$$

Furthermore, $\mathcal{E} = (\sigma B)\mathcal{E}'$ for some $\mathcal{E}'$ with $\#(\mathcal{E}') = k$ and $B = \langle X_i = f_{X_i}\rangle_{i=1}^n$.

From the two definitions for the semantics of BES, we obtain the following:

$$\begin{aligned}(\!|\mathcal{E}|\!)\theta &= (\!|\mathcal{E}'|\!)(\theta[\langle X_i\rangle_{i=1}^n = \sigma(\mathcal{F}(\mathcal{E}', B, \theta))])\\ [\![(\sigma B)\mathcal{E}']\!]\theta &= [\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n])\end{aligned}$$

From the induction hypothesis,

$$\begin{gathered}(\!|\mathcal{E}'|\!)(\theta[\langle X_i\rangle_{i=1}^n := \sigma(\mathcal{F}(\mathcal{E}', B, \theta))])\\ =\\ [\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n])\end{gathered}$$

if

$$\theta[\langle X_i\rangle_{i=1}^n := \sigma(\mathcal{F}(\mathcal{E}', B, \theta))] = \theta[\langle X_i := ((F\ (\mathcal{E}'\ B\ \theta))^n(\theta_\sigma))(X_i)\rangle_{i=1}^n]$$

Which is the case if, for all $X_i \in \mathbf{bnd}_B$,

$$(\sigma(\mathcal{F}(\mathcal{E}', B, \theta)))_i = ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)$$

Observe that we have the following.

$$\sigma(\mathcal{F}(\mathcal{E}', B, \theta)) = ((\mathcal{F}(\mathcal{E}', B, \theta))^n)(b_\sigma)$$

Therefore, we need to show that for all $X_i \in \mathbf{bnd}_B$,

$$(((\mathcal{F}(\mathcal{E}', B, \theta))^n)(b_\sigma))_i = ((F(\mathcal{E}', B, \theta))^n(\theta_\sigma))(X_i)$$

Observe that for all $1 \leq j \leq n$, $(b_\sigma)_j = \theta_\sigma(X_j)$. We will prove that, for any n tuple of Booleans b and environment θ_b such that, for all $1 \leq j \leq n$ we have $(b)_j = \theta_b(X_j)$, and for any l, the following holds.

$$\begin{gathered}(((\mathcal{F}(\mathcal{E}', B, \theta))^l)(b))_i = \\ ((F(\mathcal{E}', B, \theta))^l(\theta_b))(X_i)\end{gathered}$$

We proceed by induction on l.

$l = 0$ In this case, for any n tuple of Booleans b, any environment θ_b such that, for all $1 \leq j \leq n$ we have $(b)_j = \theta_b(X_j)$, and any $X_i \in \mathbf{bnd}_B$, we can derive the following.

$$\begin{aligned}(((\mathcal{F}(\mathcal{E}', B, \theta))^0)(b))_i &= \\ (b)_i &= \\ (\theta_b)(X_i) &= \\ ((F(\mathcal{E}', B, \theta))^0(\theta_b))(X_i)&\end{aligned}$$

$l = m + 1$ In this case, we have the following induction hypothesis; for any n tuple of Booleans b, any environment θ_b such that, for all $1 \leq j \leq n$ we have $(b)_j = \theta_b(X_j)$, and any $X_i \in \mathbf{bnd}_B$, and for any $X_i \in \mathbf{bnd}_B$, we have

$$(((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_i = ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_i) \tag{2}$$

Take some n tuple of Booleans b and environment θ_b such that, for all $1 \leq i \leq n$, $(b)_i = \theta_b(X_i)$. Furthermore, take some $X_i \in \mathbf{bnd}_B$. Then we can derive the following.

$$\begin{aligned}(((\mathcal{F}(\mathcal{E}', B, \theta))^{m+1})(b))_i &= \\ ((\mathcal{F}(\mathcal{E}', B, \theta))(((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b)))_i &= \\ [\![f_{X_i}]\!]((\!|\mathcal{E}'|\!)(\theta[\langle X_j := (((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_j\rangle_{j=1}^n]))&\end{aligned}$$

And

$$\begin{aligned}((F(\mathcal{E}', B, \theta))^{m+1}(\theta_b))(X_i) &= \\ ((F(\mathcal{E}', B, \theta))((F(\mathcal{E}', B, \theta))^m(\theta_b)))(X_i) &= \\ (||B||([\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_i)\rangle_{i=1}^n)]))(X_i) &= \\ [\![f_{X_i}]\!]([\![\mathcal{E}']\!](\theta[\langle X_i := ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_i)\rangle_{i=1}^n]))&\end{aligned}$$

Thus, we need to show that

$$\begin{gathered}[\![f_{X_i}]\!]((\!|\mathcal{E}'|\!)(\theta[\langle X_j := (((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_j\rangle_{j=1}^n])) = \\ [\![f_{X_i}]\!]([\![\mathcal{E}']\!](\theta[\langle X_j := ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_j)\rangle_{j=1}^n))\end{gathered}$$

Which is the case if

$$\begin{gathered}(\!|\mathcal{E}'|\!)(\theta[\langle X_j := (((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_j\rangle_{j=1}^n]) = \\ [\![\mathcal{E}']\!](\theta[\langle X_j := ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_j)\rangle_{j=1}^n)\end{gathered}$$

Which, according to (1), is the case if

$$\begin{gathered}\theta[\langle X_j := (((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_j\rangle_{j=1}^n] = \\ \theta[\langle X_j := ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_j)\rangle_{j=1}^n]\end{gathered}$$

Which is the case if, for all $1 \leq j \leq n$, we have

$$(((\mathcal{F}(\mathcal{E}', B, \theta))^m)(b))_j = ((F(\mathcal{E}', B, \theta))^m(\theta_b))(X_j)$$

Which follows from (2).

□

Appendix C

PBES rules

In this appendix, we explain how the proof system presented in this document could be adjusted to obtain a proof system suitable for deriving consistent consequences in the setting of PBES. For convenience, we will assume to be working in a standard setting for discussing PBES. Furthermore, we assume that the reader has some familiarity with standard definitions on PBES and the definition of consistent consequences on PBES (see also [5]).

In this appendix, when some proposition P logically implies some proposition Q, we may write $P \to Q$.

We will use a notion of autographs, which has been used to define the notion of relative consistent consequence in the setting of PBES. The autograph of a predicate variable is similar to the signature of a predicate variable, but is defined using data *terms* in stead of data *values*.

Definition C.0.1. Let $X : D_X \to \mathbb{B}$ be a predicate variable. X's autograph, denoted $\mathbf{aut}(X)$, is the product $X \times D_X$, where D_X is the data sort associated with the data type of X. We lift the notion of autographs of predicate variables to sets of predicate variables $P \subseteq \mathcal{X}$ in the natural way, so $\mathbf{aut}(P) = \bigcup_{X \in P} \mathbf{aut}(X)$. Likewise for PBES $\mathcal{E}$, we write $\mathbf{aut}(\mathcal{E})$ for $\mathbf{aut}(\mathbf{bnd}_{\mathcal{E}})$. Also, if no confusion is possible, we may write the more readable $X(d) \in \mathbf{aut}(X)$, in stead of $(X, d) \in \mathbf{aut}(X)$.

Given a relation on autographs, we define the relation on signatures induced by this relation as follows.

Definition C.0.2. Let $\mathcal{E}$ be a PBES, and let $R \subseteq \mathbf{aut}(\mathcal{E}) \times \mathbf{aut}(\mathcal{E})$ be a relation on autographs from $\mathcal{E}$. The relation on signatures *induced* by R is defined as $\bigcup_{\delta}\{(X(\llbracket d_1 \rrbracket\delta), Y(\llbracket d_2 \rrbracket\delta)) \mid (X(d_1), Y(d_2)) \in R\}$. We write $\mathbf{ind}(R)$ to indicate the relation on signatures induced by R.

Furthermore, we will use the following definitions of logical and relative consequence in the setting of predicate formulas.

Definition C.0.3. Given predicate formulas f, g, we say that g is a logical consequence of f, written $f \Rightarrow g$ if, for all data environments δ and for all predicate environments θ, we have $\llbracket f \rrbracket\theta\delta = \mathit{true}$ implies $\llbracket g \rrbracket\theta\delta = \mathit{true}$.

Definition C.0.4. Given a relation R on predicate variables, and predicate formulas f, g, we say that g is a consequence of f relative to R, written $f \overset{R}{\Rightarrow} g$

if, for all data environments δ and for all predicate environments $\theta \in \Theta_R$, we have $[\![f]\!]\theta\delta = \mathit{true}$ implies $[\![g]\!]\theta\delta = \mathit{true}$.

Definition C.0.5. Given a relation R on predicate variables, a set of Boolean data terms P and predicate formulas f, g, we say that g is a consequence of f relative to R given P, written $f \underset{P}{\overset{R}{\Rightarrow}} g$ if, for all data environments δ such that for all $b \in P$ we have $\delta(b) = \mathit{true}$ (written $\delta(P)$), and for all predicate environments $\theta \in \Theta_R$, we have $[\![f]\!]\theta\delta = \mathit{true}$ implies $[\![g]\!]\theta\delta = \mathit{true}$.

Note that we overload notation ($\Rightarrow$ and $\overset{R}{\Rightarrow}$), which is also used in the setting of propositional formulas. If it is not clear which setting is meant from the context then this will be explained.

The proof system for deriving consistent consequences on BES from this document was created by taking a sound and complete axiomatization for deriving logical consequences on propositional logic, and adding the rules CC and CNT. We will first study these two rules in Section C, and see how they could be adjusted for the setting of PBES. Since PBES uses data terms and quantifiers, while BES only use the symbols $\top$ and $\bot$, we will add some (modified) rules for handling these in Section C. We found that working with assumptions on data terms is an asset in the setting of PBES, thus we will introduce some rules to deal with these assumptions in Section C.

In Section C, we will present the resulting proof system. We theorize that this proof system allows derivations of consistent consequences between bound predicate variables from PBES, and to support this claim we will show how this proof system can derive consistent consequences between variables a PBES which cannot be solved by instantiating the PBES to a BES. Since the theory presented here is mostly conjecture, we will end with a discussion on the most critical parts where further investigation of the theory is required in Section C.

CC, CNT

The rule CC from the proof system in this document is sound because it corresponds with the the following property: For any variables X, Y bound in a BES $\mathcal{E}$, if we can derive that the right hand sides of X, Y in $\mathcal{E}$ are consequences of each other relative to $\lessdot^{\mathcal{E}}$ from an assumption that the variables are related via a consistent consequence relation, then the variables are indeed related via a consistent consequence relation.

Claim C.0.6. Given a BES $\mathcal{E}$. Then, for any bound variables X, Y with equal rank we have the following:

$$((X \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} Y) \to (f_X \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} f_Y)) \to (X \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} Y)$$

We can then straightforwardly derive the CNT rule from rule CC. When we use the rule CC, we assume that a pair of variables is related via a consistent consequence relation, and try to derive a logical consequence between the right hand sides of these variables, assuming that the variables are logical consequences. Thus, if we are in a state where we need to show that there is a logical consequence between variables, and one of our assumptions is that there is a logical consequence between these variables, then we are done.

When we try to derive consistent consequences in the setting of PBES, things become slightly more tricky. First, let's see what happens if we adjust C.0.6 to a setting of PBES.

Claim C.0.7. Let $\mathcal{E}$ be a PBES, and take some variables X, Y bound in $\mathcal{E}$ with equal ranks. Furthermore, let x, y be data terms such that $X(x)$ and $Y(y)$ are elements from $\mathbf{aut}(\mathcal{E})$. Then we have the following:

$$((X(x) \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} Y(y)) \to (f_X(x) \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} f_Y(y))) \to (X(x) \overset{\lessdot^{\mathcal{E}}}{\Longrightarrow} Y(y))$$

Creating rules CC and CNT based on C.0.7 would allow us to derive consistent consequences between specific elements from the autograph of a PBES. In some cases this might be sufficient, in particular if we can all ready solve the PBES by instantiating it to a BES and solving the BES. However, let's say that we have some PBES with some bound variables X, Y, Z, such that the domain of X is the natural numbers. We might be interested in relating $X(n)$ to $Y(y)$ for some y if n is even, and $X(n)$ to $Z(z)$ if n is odd. However, using C.0.7 to define CC (and CNT), we then would not be able to use the fact that if we assume that $X(n) \lessdot^{\mathcal{E}} Y(y)$ for even n, we also have $X(k+2) \lessdot^{\mathcal{E}} Y(y)$ for some even k, unless $n = k+2$.

Therefore, we wish to generalize this property, such that we can work with assumptions on data terms and we are trying to derive consistent consequences between sets of the autographs of predicate variables, in stead of deriving consistent consequences between specific elements from the autographs of predicate variables. We propose the following adjustment of C.0.7, which should be sufficiently powerful to derive consistent consequence between sets of autographs.

Claim C.0.8. Let $\mathcal{E}$ be a PBES, and take some variables X, Y bound in $\mathcal{E}$ with equal ranks. Furthermore, let a, b be data terms such that $X(a) \in \mathbf{aut}(\mathcal{E})$ and $Y(b) \in \mathbf{aut}(\mathcal{E})$, and let $P(a, b)$ be a set of Boolean data terms over a and b.

We have that, if the following holds:

$$\begin{gathered}\forall x, y : (X(x) \in Aut_{\mathcal{E}} \wedge Y(y) \in Aut_{\mathcal{E}}) \to \\ (X(x) \xrightarrow[P(x,y)]{\lessdot^{\mathcal{E}}} Y(y) \to (f_X(x) \xrightarrow[P(x,y)]{\lessdot^{\mathcal{E}}} f_Y(y)))\end{gathered}$$

Then we can conclude that, for all x, y such that $X(x) \in Aut_{\mathcal{E}}$, $Y(y) \in Aut_{\mathcal{E}}$, and for all data environments δ such that $P(x, y)$ holds on all its Boolean data terms, then we have $X(\delta(x)) \lessdot^{\mathcal{E}} Y(\delta(y))$.

Using this property, we can use P to express conditions on variables used in data terms. The CC rule corresponding with C.0.8 could look as follows.

$$\frac{P \mathbin{\&} \Gamma, (\{X(x) \subset Y(y) \mid P\})[\overline{x,y} := \overline{a}] \vdash f_X(x) \subset f_Y(y)}{P \mathbin{\&} \Gamma \vdash X(x) \subset Y(y)} *$$

Here, $\overline{x,y}$ is the set of data variables which occur in data terms x, y, and $(\{X(x) \subset Y(y) \mid P\})[\overline{x,y} := \overline{a}]$ stands for the simultaneous replacement of all variables from x, y in $X(x)$, $Y(y)$ and P with variables from $\overline{a}$.

*: This rule should be applied with the following restrictions. First, X and Y should be bound in the PBES on which one wishes to derive consistent consequences, and their ranks should be equal. Furthermore, $\overline{a}$ is a set of fresh variables sufficiently large such that all variables from x and y are replaced with variables from a and variables from x, y with the same name should be replaced with the same variable from $\overline{a}$.

The idea behind this notation is that, in Γ we are building a set of predicate variables with data elements which we assume to be consistent consequences, and on the right hand side of $\vdash$ we need to show that two predicate formulas are consistent consequences for all data environments such that all Boolean data terms in P hold.

Within BES, the CNT rule was straightforward. Given the previous CC rule, derived from the property in C.0.7, we can also present a CNT rule.

$$\frac{}{P \mathbin{\&} \Gamma \vdash X(x) \subset Y(y)} \; \{X(x) \subset Y(y) \mid P\} \subseteq \Gamma$$

Here, $\{X(x) \subset Y(y) \mid P\} \subseteq \Gamma$ *means* $\bigcup_{\delta}\{(X(\delta(x)), Y(\delta(y))) \mid \delta(P)\}$ *is a subset of* $\bigcup_{\delta,\{X(x) \subset Y(y) \mid P\} \in \Gamma}\{(X(\delta(x)), Y(\delta(y))) \mid \delta(P)\}$.

The idea is that, if what we wish to derive follows from our assumptions, then we are done. However, there is a significant downside to this rule, since we have to show that something is a subset of Γ. It would be interesting to investigate how the rules can be changed such that we do not have to show subsets.

Data and quantifiers

Changing the CC and CNT rules in the proof system for BES with these new rules on PBES, is insufficient for creating a proof system powerful enough for deriving consistent consequences on PBES. At least, the predicate formulas used in PBES have a more complex structure than the propositional formulas in BES. Thus, we need ways to handle these differences; we need to be able to handle data terms (and in particular Boolean data terms) and quantifiers.

In the setting for BES we had the primitives $\top$ and $\bot$. However, in the setting for PBES, we have arbitrary Boolean data terms, whose semantics are given by data environments. Since P restricts the set of data environments we are considering, given a Boolean data term b, if all data environments δ such that $\delta(P)$ holds have $\delta(b) = \mathit{true}$ (written $P \to b$), then from our assumptions we know that b can be used as we used $\top$ in the setting for BES. Similarly, if all data environments δ such that $\delta(P)$ holds have $\delta(b) = \mathit{false}$ (written $P \to \neg b$), then from our assumptions we know that b can be used as we used $\bot$ in the setting for BES. Thus, we change the rules for $\top$ and $\bot$ as follows.

$$\text{TOP} \quad \frac{}{P \mathbin{\&} \Gamma \vdash \alpha \subset \alpha \wedge b} \; P \to b \qquad \text{BOT} \quad \frac{}{P \mathbin{\&} \Gamma \vdash \alpha \vee b \subset \alpha} \; P \to \neg b$$

Let P be a set of Boolean data terms. Furthermore, let t_1, t_2 be two data terms of the same type. We will write $P \to t_1 = t_2$ if, for all data environments δ such that $P(\delta)$ holds, we have $\delta(t_1) = \delta(t_2)$. If two predicate formulas are equal up to some data terms t_1 and t_2, and we know from our assumptions that these two data rules are equal, then the two predicate formulas are equal under our assumptions. Thus, the reflexivity rule is modified to allow equivalent data terms. This results in the following rule.

$$\text{REF} \quad \frac{}{P \;\&\; \Gamma \vdash \alpha[t := t_1] \subset \alpha[t := t_2]}\; P \to t_1 = t_2$$

For the quantifier rules, we can simply copy the rules used in the sequent calculus LK, as defined by Gentzen [11], and add them to our proof system.

$$\forall_L \quad \frac{P \;\&\; \Gamma \vdash \alpha[d := d'] \subset \beta}{P \;\&\; \Gamma \vdash \forall d : D.\alpha \subset \beta} \qquad \exists_L \quad \frac{P \;\&\; \Gamma \vdash \alpha[d := z] \subset \beta}{P \;\&\; \Gamma \vdash \exists d : D.\alpha \subset \beta}$$

$$\forall_R \quad \frac{P \;\&\; \Gamma \vdash \alpha \subset \beta[d := z]}{P \;\&\; \Gamma \vdash \alpha \subset \forall d : D.\beta} \qquad \exists_R \quad \frac{P \;\&\; \Gamma \vdash \alpha \subset \beta[d := d']}{P \;\&\; \Gamma \vdash \alpha \subset \exists d : D.\beta}$$

Here, d and d' are data terms of type D. Furthermore, z is a fresh data variable of type D, i.e. z does not occur anywhere below the horizontal line.

Manipulating P

Currently, we have no way of manipulating the assumptions on data terms stored in P. Consider the following statement in a proof tree (in the setting of some PBES).

$$P \;\&\; \Gamma \;\vdash\; b \wedge \alpha \subset \beta \tag{3}$$

Where P is a set of Boolean data terms, Γ is a relation on elements from the autographs of a PBES, b is a Boolean data term, and α and β are predicate formulas. Then we mean the following. If $\mathbf{ind}(\Gamma)$ is a set of signatures which are related via a consistent consequence relation on the PBES we are interested in, then for all data environments δ such that $\delta(P)$ holds, $\delta(\beta)$ is a consistent consequence of $\delta(b \wedge \alpha)$, i.e.

$$\mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}} \to (b \wedge \alpha) \overset{\lessdot^{\mathcal{E}}}{\underset{P}{\Longrightarrow}} \beta$$

We can derive the following:

$$\mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}} \rightarrow (b \wedge \alpha) \overset{\lessdot^{\mathcal{E}}}{\underset{P}{\Longrightarrow}} \beta$$
$$\leftrightarrow$$
$$\forall \delta . (\delta(P) \wedge \mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}}) \rightarrow (\forall \theta \in \Theta_{\lessdot^{\mathcal{E}}} . [\![b \wedge \alpha]\!] \theta\delta \rightarrow [\![\beta]\!] \theta\delta)$$
$$\leftrightarrow$$
$$\forall \delta . (\delta(P, b) \wedge \mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}}) \rightarrow (\forall \theta \in \Theta_{\lessdot^{\mathcal{E}}} . [\![\alpha]\!] \theta\delta \rightarrow [\![\beta]\!] \theta\delta)$$
$$\leftrightarrow$$
$$\mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}} \rightarrow \alpha \overset{\lessdot^{\mathcal{E}}}{\underset{P,b}{\Longrightarrow}} \beta$$

Thus, for deriving (3), it is sufficient if we can derive the following.

$$P, b \ \& \ \Gamma \ \vdash \ \alpha \subset \beta$$

Therefore we can add the following rule for adding Boolean data terms from the antecedant to P, which we will cal DAT1.

$$\frac{P, b \ \& \ \Gamma \vdash \alpha \subset \beta}{P \ \& \ \Gamma \vdash b \wedge \alpha \subset \beta}$$

Next, consider the following statement in a proof tree.

$$P \ \& \ \Gamma \ \vdash \ \alpha \subset \beta \vee b \tag{4}$$

Again, P is a set of Boolean data terms, Γ is a relation on elements from the autographs of a PBES, b is a Boolean data term, and α and β are predicate formulas.

Using a similar reasoning, we can add the following rule for adding Boolean data terms from the consequent to P, which we will cal DAT2.

$$\frac{P, \neg b \ \& \ \Gamma \vdash \alpha \subset \beta}{P \ \& \ \Gamma \vdash \alpha \subset \beta \vee b}$$

We can also modify P when we are eliminating quantifiers. Consider the following statement in a proof tree (in the setting of some PBES).

$$P \ \& \ \Gamma \ \vdash \ \forall d : D . \alpha \subset \beta \tag{5}$$

Where P is a set of Boolean data terms, Γ is a relation on elements from the autographs of a PBES, and d is a data variable of type d. Then we mean the following.

$$\mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}} \rightarrow (\forall d : D . \alpha) \overset{\lessdot^{\mathcal{E}}}{\underset{P}{\Longrightarrow}} \beta$$

Which is equivalent to the following.

$$\forall \delta.(\delta(P) \wedge \mathbf{ind}(\Gamma) \subseteq \lessdot^{\mathcal{E}}) \rightarrow \forall \theta \in \Theta_{\lessdot^{\mathcal{E}}}.[\![\forall d : D.\alpha]\!]\theta\delta \rightarrow [\![\beta]\!]\theta\delta \tag{6}$$

Let $\mathbf{d}$ be some data variable of type D which does not occur in α, β, P or Γ, and let b be a Boolean data term such that the only data variable which occurs in b is $\mathbf{d}$, and such that we do not have $P \rightarrow \neg b$ (i.e. b does not contradict our assumptions in P). Then (6) holds if the following holds:

$$\forall \delta : (\delta(P \cup \{b\}) \wedge \Gamma \subseteq \lessdot^{\mathcal{E}}) \rightarrow \forall \theta \in \Theta_{\lessdot^{\mathcal{E}}}.[\![\alpha[d := \mathbf{d}]\!]\theta\delta) \rightarrow [\![\beta]\!]\theta\delta$$

Thus, for deriving (5), it is sufficient if we can derive the following.

$$P, b \;\&\; \Gamma \;\vdash\; \alpha[d := \mathbf{d}] \subset \beta$$

Therefore we can add the following rule for adding Boolean data terms from the antecedant to P, which we will cal $\mathrm{DAT}_{\forall}$.

$$\frac{P, b \;\&\; \Gamma \vdash \alpha[d := \mathbf{d}] \subset \beta}{P \;\&\; \Gamma \vdash \forall d : D.\alpha \subset \beta} \, *$$

**: b is a Boolean data term such that not $P \rightarrow \neg b$ and the only variable which occurs in b is $\mathbf{d}$.*

We can apply a similar reasoning for existential quantifiers in the consequent, to obtain the following rule which we will call $\mathrm{DAT}_{\exists}$.

$$\frac{P, b \;\&\; \Gamma \vdash \alpha \subset \beta[d := \mathbf{d}]}{P \;\&\; \Gamma \vdash \alpha \subset \exists d : D.\beta} \, *$$

**: b is a Boolean data term such that not $P \rightarrow \neg b$ and the only variable which occurs in b is $\mathbf{d}$.*

A proof system for consistent consequence on PBES

In Table C.1 a proof system for deriving consistent consequences on predicate variables from a PBES is shown. The rules are as discussed in the previous sections. We believe that this system provides a basis for future research into a more complete proof system for PBES. To show that this proof system is indeed capable of deriving consistent consequences on PBES, we will show how the proof system can be used to derive consistent consequences on a PBES, which cannot be instantiated into BES.

Example C.0.9. Consider the following PBES.

$$\begin{array}{l}
(\nu\langle(A_{\top} = B_{\top}) \\
\quad (A_{\bot} = B_{\bot}) \\
\quad (X(n : N) = Y(n))\rangle) \\
(\mu\langle(B_{\top} = A_{\bot}) \\
\quad (B_{\bot} = B_{\bot} \vee B_{\top}) \\
\quad (Y(n : N) = (\mathrm{even}(n) \wedge X(n+1)) \vee \\
\qquad (\mathrm{odd}(n) \wedge (Y(n) \vee Y(n+1)))\rangle)
\end{array}$$

This PBES (which we will refer to as $\mathcal{E}$) is the result of merging the PBES containing the equations for $A_\top, A_\bot, B_\top, B_\bot$ (which we will refer to as $\mathcal{E}^1$) with the PBES containing the equations for X and Y (which we will refer to as $\mathcal{E}^2$).

The solution for PBES $\mathcal{E}^1$ can be easily found by instantiating to BES (it is a BES). However, PBES $\mathcal{E}^2$ cannot be solved by instantiating to BES, due to the presence of an infinite chain of dependencies.

$$(\nu\langle(A_\top = B_\top)(A_\bot = B_\bot)(X_0 = Y_0)(X_1 = Y_1)\cdots\rangle)$$
$$(\mu\langle(B_\top = A_\bot)(B_\bot = B_\bot \vee B_\top)(Y_0 = X_1)(Y_1 = Y_1 \vee Y_2)\cdots\rangle)$$

However, for even v we have $A_\top \lessdot^{\mathcal{E}} X(v)$, and for odd v we have $A_\bot \lessdot^{\mathcal{E}} X(v)$. Likewise, for even v we have $B_\top \lessdot^{\mathcal{E}} Y(v)$, and for odd v we have $B_\bot \lessdot^{\mathcal{E}} Y(v)$ (see also [5]).

Since any solution of $\mathcal{E}^1$ assigns *true* to all predicate variables in $\mathcal{E}^1$, we also know that the solution for any n of $X(n)$ and $Y(n)$ in $\mathcal{E}^2$ must be *true*.

In Figure C.1, a proof tree is shown tree for deriving $A_\top \lessdot^{\mathcal{E}} X(v)$ for even v. The numbers are as indicated below. For convenience, we assume that we can derive rules similar to the following derived rules cSPLIT, cGROW$_L$, cGROW$_R$, aSPLIT, aGROW$_L$ and aGROW$_R$ from the system for BES.

In the following, let Γ be $\{A_\top \subset X(a) \mid \text{even}(a)\}, \{B_\top \subset Y(b) \mid \text{even}(b)\}$ and let Γ' be $\{A_\bot \subset X(c+1) \mid \text{even}(c)\}, \{B_\bot \subset Y(d+1) \mid \text{even}(d)\}$. Also, let P be even(v).

1. $P \,\&\, \Gamma \vdash A_\bot \subset A_\bot \wedge \text{even}(v)$
2. $P \,\&\, \Gamma \vdash A_\bot \subset \text{even}(v) \wedge A_\bot$
3. $P \,\&\, \Gamma \vdash \text{even}(v) \wedge A_\bot \subset \text{even}(v)$
4. $P \,\&\, \Gamma \vdash A_\bot \subset \text{even}(v)$
5. $P \,\&\, \Gamma \vdash A_\bot \subset \text{even}(v)$
6. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \vee B_\top \subset (B_\bot \vee B_\top) \wedge \text{odd}(v+1)$
7. $P \,\&\, \Gamma, \Gamma' \vdash (B_\bot \vee B_\top) \wedge \text{odd}(v+1) \subset \text{odd}(v+1) \wedge (B_\bot \vee B_\top)$
8. $P \,\&\, \Gamma, \Gamma' \vdash \text{odd}(v+1) \wedge (B_\bot \vee B_\top) \subset \text{odd}(v+1)$
9. $P \,\&\, \Gamma, \Gamma' \vdash (B_\bot \vee B_\top) \wedge \text{odd}(v+1) \subset \text{odd}(v+1)$
10. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \vee B_\top \subset \text{odd}(v+1)$
11. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \subset Y(v+1)$
12. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \subset Y(v+1) \vee Y(v+1+1)$
13. $P \,\&\, \Gamma, \Gamma' \vdash B_\top \subset Y(v+1+1)$
14. $P \,\&\, \Gamma, \Gamma' \vdash B_\top \subset Y(v+1) \vee Y(v+1+1)$
15. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \vee B_\top \subset Y(v+1) \vee Y(v+1+1)$
16. $P \,\&\, \Gamma, \Gamma' \vdash B_\bot \vee B_\top \subset \text{odd}(v+1) \wedge (Y(v+1) \vee Y(v+1+1))$

17. $P \ \& \ \Gamma, \Gamma' \ \vdash B_{\perp} \vee B_{\top} \subset (\text{even}(v+1) \wedge X(v+1+1)) \vee (\text{odd}(v+1) \wedge (Y(v+1) \vee Y(v+1+1)))$

18. $P \ \& \ \Gamma, \{A_{\perp} \subset X(c+1) \mid \text{even}(c)\} \ \vdash B_{\perp} \subset Y(v+1)$

19. $P \ \& \ \Gamma \ \vdash A_{\perp} \subset X(v+1)$

20. $P \ \& \ \Gamma \ \vdash A_{\perp} \subset \text{even}(v) \wedge X(v+1)$

21. $P \ \& \ \Gamma \ \vdash A_{\perp} \subset (\text{even}(v) \wedge X(v+1)) \vee (\text{odd}(v) \wedge (Y(v) \vee Y(v+1)))$

22. $P \ \& \ \{A_{\top} \subset X(a) \mid \text{even}(a)\} \vdash B_{\top} \subset Y(v)$

23. $P \ \& \ \vdash A_{\perp} \subset X(v)$

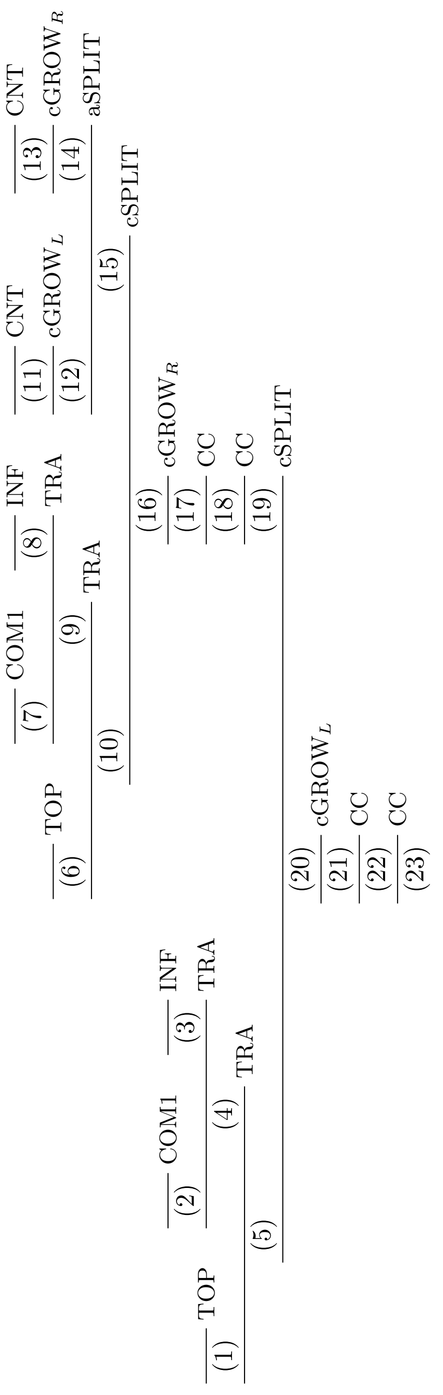

Figure C.1: A proof tree for deriving $A_\top \lessdot^{\mathcal{E}} X(v)$ for even v from Example C.0.9.

Table C.1: Proof system for consistent consequences on PBES $\mathcal{E}$; α, β and γ are arbitrary predicate formulas; X, Y are predicate variables; P is a set of Boolean (data) expressions; x and d_X are data terms of data sort D_X; y is a data term of data sort D_Y; d, $\mathbf{d}$ and z are data variables of data sort D; d' is a data term of data sort D; b and $\mathbf{b}$ are Boolean data terms; t, t_1, t_2 are data terms of the same data sort; $\mathbf{d}$ and z are free variables; the only variable occurring in $\mathbf{b}$ is $\mathbf{d}$; $\bar{a}$ is a set of free variables

Axioms:

The same as in the proof system for BES, but without TOP and BOT.

Logic rules:

REF $$\frac{}{P \;\&\; \Gamma \vdash \alpha[t := t_1] \subset \alpha[t := t_2]}\; P \rightarrow t_1 = t_2$$

TRA $$\frac{P \;\&\; \Gamma \vdash \alpha \subset \beta \qquad P \;\&\; \Gamma \vdash \beta \subset \gamma}{P \;\&\; \Gamma \vdash \alpha \subset \gamma}$$

CTX $$\frac{P \;\&\; \Gamma \vdash \alpha(d_X) \subset \beta(d_X)}{P \;\&\; \Gamma \vdash \gamma[X := \lambda d_X : D_X.\alpha(d_X)] \subset \gamma[X := \lambda d_X : D_X.\beta(d_X)]}$$

Data rules:

TOP $$\frac{}{P \;\&\; \Gamma \vdash \alpha \subset \alpha \wedge b}\; P \rightarrow b$$

BOT $$\frac{}{P \;\&\; \Gamma \vdash \alpha \vee b \subset \alpha}\; P \rightarrow \neg b$$

DAT1 $$\frac{b, P \;\&\; \Gamma \vdash \alpha \subset \beta}{P \;\&\; \Gamma \vdash b \wedge \alpha \subset \beta}$$

DAT2 $$\frac{\neg b, P \;\&\; \Gamma \vdash \alpha \subset \beta}{P \;\&\; \Gamma \vdash \alpha \subset b \vee \beta}$$

Quantifier rules:

$\forall_L$ $$\frac{P \;\&\; \Gamma \vdash \alpha[d := d'] \subset \beta}{P \;\&\; \Gamma \vdash \forall d : D.\alpha \subset \beta}$$

$\exists_L$ $$\frac{P \;\&\; \Gamma \vdash \alpha[d := z] \subset \beta}{P \;\&\; \Gamma \vdash \exists d : D.\alpha \subset \beta}\; z \text{ fresh}$$

$\forall_R$ $$\frac{P \;\&\; \Gamma \vdash \alpha \subset \beta[d := z]}{P \;\&\; \Gamma \vdash \alpha \subset \forall d : D.\beta}\; z \text{ fresh}$$

$\exists_L$ $$\frac{P \;\&\; \Gamma \vdash \alpha \subset \beta[d := d']}{P \;\&\; \Gamma \vdash \alpha \subset \exists d : D.\beta}$$

DAT$_\forall$ $$\frac{P, \mathbf{b} \;\&\; \Gamma \vdash \alpha[d := \mathbf{d}] \subset \beta}{P \;\&\; \Gamma \vdash \forall d : D.\alpha \subset \beta}$$

DAT$_\exists$ $$\frac{P, \mathbf{b} \;\&\; \Gamma \vdash \alpha[d := \mathbf{d}] \subset \beta}{P \;\&\; \Gamma \vdash \forall d : D.\alpha \subset \beta}$$

Consistent Consequence rules:

CC $$\frac{P \;\&\; \Gamma, (\{X(x) \subset Y(y) \mid P\})[\overline{x, y} := \bar{a}] \vdash f_X(x) \subset f_Y(y)}{P \;\&\; \Gamma \vdash X(x) \subset Y(y)}\; *$$

CNT $$\frac{}{P \;\&\; \Gamma \vdash X(x) \subset Y(y)}\; \{X(x) \subset Y(y) \mid P\} \subseteq \Gamma$$

$*$:$\boldsymbol{rank}(X) = \boldsymbol{rank}(Y)$ and $X, Y \in \boldsymbol{bnd}_{\mathcal{E}}$

What's next for this system

We proposed a proof system in this appendix and have shown that it can be used to derive consistent consequences. However, due to time constraints the soundness of this proof system has not been checked thoroughly, as well as some other details. To show soundness of the proof system, the following claim should be proven:

Claim C.0.10. Given a PBES $\mathcal{E}$ and $X(x), Y(y) \in \mathbf{aut}(\mathcal{E})$. If we can build a proof tree with $P \; \& \; \vdash X(x) \subset Y(y)$ as the root, then we have

$$\forall \delta . \delta(P) \rightarrow X(\delta(x)) \lessdot^{\mathcal{E}} Y(\delta(y))$$

Furthermore, we have given a few rules for adding expressions to P, but it remains to be seen if these are sufficient, and if it would be desirable to also be able to modify expressions from P (for example strengthening or weakening expressions). Both $\mathrm{DAT}_\forall$ and $\mathrm{DAT}_\exists$ should be investigated since these rules are currently fairly strongly formulated ($\mathbf{d}$ needs to be a free variable, $\mathbf{b}$ can only use $\mathbf{d}$). It is probable that some relaxations can be made here.

Also, the CNT rule is not fully a syntactic rule yet. It would be desirable to add or modify rules such that we are not dealing with determining subsets when applying the CNT rule. This would probably require adding rules capable of manipulating (strengthening or weakening) Γ.

Furthermore, for the CC rule we made the very safe choice of renaming all variables in the set $\{X(x) \subset Y(y) \mid P\}$ which is added to Γ. This is probably a bit to safe, and relaxing this requirement would make the system more usable.

The theory outside of the proof system also warrants further investigation. We implicitly assumed some properties of relative consequence given a set of assumptions to be similar to properties of relative consequence, but the relation between the two notions should be further explored. We also used a notion of (relative) consistent consequence (both on predicate formulas and on predicate variables) without giving a formal definition. These definitions and properties will probably be required in some form for formally checking the soundness of the proof system.

Considering all these remaining questions, it is highly unlikely that the system is complete. Completeness of the proof system for BES, which we have shown in this document, was achieved by showing that we can build the consistent consequence relation at the left-hand side of $\vdash$. To be able to mimic this method for proving this system complete, it would first need to be investigated if all consistent consequence relations can be finitely represented by the notation we use: $\{X(x) \subset Y(y) \mid P\}$.

Appendix D

The remaining parts of Coq

In this appendix, we provide descriptions of the lemmas in the Coq files accompanying this report, which were not discussed in Chapter 6.

6.1

Lemma D.0.1. propForm_solution_decidable
Given a propositional formula and an environment, the semantics of this propositional formula under this environment is decidable:

$$\forall f, \forall \theta, [\![f]\!]\theta \vee \neg [\![f]\!]\theta$$

Lemma D.0.2. propForm_solution_deMorgan_conj
A version of De Morgan's law on the semantics of propositional formulas.

$$\forall f_1 f_2, \forall \theta, \neg([\![f_1]\!]\theta \wedge [\![f_2]\!]\theta) \rightarrow (\neg [\![f_1]\!]\theta \vee \neg [\![f_2]\!]\theta)$$

Lemma D.0.3. environment_intersect_distributes_neg
If the solution of two propositional formulas under two environments is *false*, then the solution of the propositional formulas under the intersection of the environments is also *false*.

$$\forall \theta_1 \theta_2, \forall f_1 f_2, (\neg [\![f_1]\!]\theta_1 \wedge \neg [\![f_2]\!]\theta_2) \rightarrow (\neg [\![f_1]\!](\theta_1 \| \theta_2) \wedge \neg [\![f_2]\!](\theta_1 \| \theta_2))$$

Lemma D.0.4. maxlteq
For any natural numbers x, y, z such that $x \leq y$ or $x \leq z$, we have that x is less than or equal to the maximum of y and z

$$\forall x\ y\ z, (x \leq y \vee x \leq z) \rightarrow x \leq (max(y, z))$$

Lemma D.0.5. uses_lteq_mxUsed
For any propositional formula f and propositional variable x occurring in f, x is less than or equal to the largest variable used in f

$$\forall f, \forall x, uses\ f\ x \rightarrow x \leq (mxUsed\ f)$$

Lemma D.0.6. suc_mxUsed_unused
For any propositional formula f, the propositional variable encoded by the successor of the largest propositional variable used in f is not used in f

$$\forall f, \neg(\mathit{uses}\ f\ (S(\mathit{mxUsed}\ f)))$$

Lemma D.0.7. unused_no_rewrite
For any propositional formulas f, f', and propositional variable x, if x is not used in f then rewriting x in f to f' results in f (with no changes)

$$\forall f f', \forall x, (\neg \mathit{uses}\ f\ x) \to f[x := f'] = f$$

Lemma D.0.8. environment_union_grows
Given two environments, and a propositional formula, if either environment gives the solution *true* to the propositional formula, then the union of the environments solves the propositional formula to *true* as well.

$$\forall f, \forall \theta_1 \theta_2, ([\![f]\!]\theta_1 \vee [\![f]\!]\theta_2) \to [\![f]\!](\theta_1 \| \theta_2)$$

Lemma D.0.9. environment_point_minimal
Given an environment assigning *true* to a propositional variable, for any propositional formula, if this environment solves this formula to *false* then the propositional formula also solves to false under the environment point of x.

$$\forall f, \forall x, \forall \theta, \theta(x) \to \neg[\![f]\!]\theta \to \neg[\![f]\!](\mathit{environment\_point}\ x)$$

Lemma D.0.10. environment_union_distributes
If the solution of two propositional formulas under two environments is *true*, then the solution of both propositional formulas under the union of the environments is also *true*.

$$\forall \theta_1 \theta_2, \forall f_1 f_2, ([\![f_1]\!]\theta_2 \wedge [\![f_2]\!]\theta_2) \to ([\![f_1]\!](\theta_1 \| \theta_2) \wedge [\![f_2]\!](\theta_1 \| \theta_2))$$

Lemma D.0.11. exists_unused
For any propositional formula f, there exists a propositional variable which is not used in f

$$\forall f, \exists x, \neg(\mathit{uses}\ f\ x)$$

6.1.1

Lemma D.0.12. propForm_eqv_bot
If a propositional formula f evaluates to *false* regardless of the environment, then f is equivalent to $\bot$.

$$\forall f, (\forall \theta, \neg[\![f]\!]\theta) \leftrightarrow f \Leftrightarrow \bot$$

Lemma D.0.13. eqv_bot_dec
Any propositional formula f is either equivalent to *false* under all evaluations, or there exists an environment for which f evaluates to *true*

$$\forall f, ((\forall \theta, \neg [\![f]\!]\theta) \vee \exists \theta, [\![f]\!]\theta)$$

Lemma D.0.14. eqv_propForm_bot_decidable
It is decidable if a propositional formula g is equivalent to $\bot$.

$$\forall g, g \Leftrightarrow \bot \vee \neg(g \Leftrightarrow \bot)$$

Lemma D.0.15. neq_var_bot
No variable is equivalent to $\bot$.

$$\forall x, \neg((var\ x) \Leftrightarrow \bot)$$

Lemma D.0.16. propForm_eqv_top
A propositional formula f evaluates to *true* under all environments iff f is equivalent to $\top$

$$\forall f, (\forall \theta, [\![f]\!]\theta) \leftrightarrow f \Leftrightarrow \top$$

Lemma D.0.17. not_top_get_not_true
If a propositional formula f is not equivalent to $\top$ then there exists an environment for which f evaluates to *false*

$$\forall f, (\neg(\top \Leftrightarrow f)) \to \exists \theta, \neg [\![f]\!]\theta$$

Lemma D.0.18. eqv_top_decidable
Any propositional formula f is either equivalent to $\top$ or not

$$\forall f, f \Leftrightarrow \top \vee \neg(f \Leftrightarrow \top)$$

Lemma D.0.19. neq_var_top
No variable is equivalent to $\top$

$$\forall x, \neg((var\ x) \Leftrightarrow \top)$$

Lemma D.0.20. var_cons_propForm_decidable
For any propositional formula f and any propositional variable x, either x is a logical consequence of f or not

$$\forall f, \forall x, f \Rightarrow (var\ x) \vee \neg(f \Rightarrow (var\ x))$$

Lemma D.0.21. neqv_bot_full_true
If a propositional formula f is not equivalent to $\perp$, then f evaluates to *true* under the θ_ν.

$$\forall f, (\neg(f \Leftrightarrow \perp)) \rightarrow [\![f]\!]\theta_\nu$$

Lemma D.0.22. full_false_eqv_bot
If a propositional formula f evaluates to *false* under the θ_ν then f is not equivalent to $\perp$.

$$\forall f, \neg[\![f]\!]\theta_\nu \rightarrow f \Leftrightarrow \perp$$

Lemma D.0.23. eqv_bot_full_false
If a propositional formula f is equivalent to $\perp$ then f evaluates to *false* under the θ_ν

$$\forall f, f \Leftrightarrow \perp \rightarrow \neg[\![f]\!]\theta_\nu$$

Lemma D.0.24. empty_true_eqv_top
If a propositional formula f evaluates to *true* under the θ_μ then f is equivalent to $\top$

$$\forall f, [\![f]\!]\theta_\mu \rightarrow f \Leftrightarrow \top$$

Lemma D.0.25. cons_var_propForm
For any propositional formula f and any propositional variable x, either f is a logical consequence of x or there exists an environment for which x evaluates to *true* and f evaluates to *false*

$$\forall f, \forall x, (\forall \theta, \theta(x) \rightarrow [\![f]\!]\theta) \vee (\exists \theta, \theta(x) \wedge \neg[\![f]\!]\theta)$$

Lemma D.0.26. cons_propForm_var
For any propositional formula f and any propositional variable x, either x is a logical consequence of f or there exists an environment for which f evaluates to *true* and x evaluates to *false*.

$$\forall f, \forall x, (\forall \theta, [\![f]\!]\theta \rightarrow \theta(x)) \vee (\exists \theta, [\![f]\!]\theta \wedge \neg\theta(x))$$

6.1.2

Lemma D.0.27. distribute_f_top
Distribute called on $\top$ and some propositional formula f returns a formula equivalent to f.

$$\forall f, \forall \theta, [\![f]\!]\theta \leftrightarrow [\![dist(f)(\top)]\!]\theta$$

Lemma D.0.28. distribute_top_f
Distribute called on some propositional formula f and $\top$ returns a formula equivalent to f.

$$\forall f, \forall \theta, [\![f]\!]\theta \leftrightarrow [\![\mathit{dist}(\top)(f)]\!]\theta$$

Lemma D.0.29. distribute_bot_f
Distribute called on $\perp$ and some propositional formula f returns a formula equivalent to $\perp$.

$$\forall f, \forall \theta, [\![\mathit{dist}(\perp)(f)]\!]\theta \leftrightarrow \mathit{False}$$

Lemma D.0.30. distribute_f_bot
Distribute called on $\perp$ and some propositional formula f returns a formula equivalent to $\perp$.

$$\forall f, \forall \theta, [\![\mathit{dist}(f)(\perp)]\!]\theta \leftrightarrow \mathit{False}$$

Lemma D.0.31. distribute_f_var
Distribute called on some propositional formula f and some propositional variable X returns a formula equivalent to $f \wedge X$.

$$\forall f, \forall X, \forall \theta, [\![\mathit{dist}(f)(\mathit{var}X)]\!]\theta \leftrightarrow [\![f \wedge_p (\mathit{var}X)]\!]\theta$$

Lemma D.0.32. distribute_var_f
Distribute called on some propositional variable X and some propositional formula f returns a formula equivalent to $X \wedge f$.

$$\forall f, \forall X, \forall \theta, [\![\mathit{dist}(\mathit{var}X)(f)]\!]\theta \leftrightarrow [\![(\mathit{var}X) \wedge_p f]\!]\theta$$

Lemma D.0.33. distribute_DNF_left
Given two DNF formulas f, g, f is a logical consequence of the result of calling distribute on f and g.

$$\forall f g, \mathit{DNF}(f) \to \mathit{DNF}(g) \to \mathit{dist}(f)(g) \Rightarrow f.$$

Lemma D.0.34. distribute_DNF_right
Given two DNF formulas f, g, g is a logical consequence of the result of calling distribute on f and g.

$$\forall f g, \mathit{DNF}(f) \to \mathit{DNF}(g) \to \mathit{dist}(f)(g) \Rightarrow g.$$

Lemma D.0.35. env_intersect_conj
For any clause f and any pair of environments θ_1 and θ_2, if $[\![f]\!]\theta_1 = \mathit{true}$ and $[\![f]\!]\theta_2 = \mathit{true}$ then $[\![f]\!](\theta_1 \& \theta_2) = \mathit{true}$.

$$\forall f, \mathit{clause}(f) \to \forall \theta_1 \theta_2, [\![f]\!]\theta_1 \to [\![f]\!]\theta_2 \to [\![f]\!](\theta_1 \& \theta_2)$$

6.1.3

Lemma D.0.36. conj_full_ncons_max_point
Given a clause f which evaluates to *true* under the *full_environment*, for any variable x, if x is not a logical consequence of f, then f evaluates to *true* under the environment assigning *true* to everything but x

$$\begin{aligned}&\forall f, [\![f]\!]\theta_\nu \to \mathit{clause}(f) \to \\ &\quad \forall x, \neg(f \Rightarrow (\mathit{var}\ x)) \to [\![f]\!](\mathit{environment\_max\_point}\ x)\end{aligned}$$

Lemma D.0.37. conj_cons_propForm_decidable
Given a clause f and a propositional formula g, either g is a logical consequence of f, or there exists an environment for which f evaluates to *true* and g evaluates to *false*

$$\forall f g, \mathit{clause}(f) \to (f \Rightarrow g \vee \exists \theta, ([\![f]\!]\theta \wedge \neg [\![g]\!]\theta))$$

Lemma D.0.38. DNF_cons_propForm_decidable
For any DNF formula f and propositional formula g, either g is a logical consequence of f, or there exists an environment for which f evaluates to *true* and g evaluates to *false*

$$\forall f g, \mathit{DNF}(f) \to (f \Rightarrow g \vee \exists \theta, ([\![f]\!]\theta \wedge \neg [\![g]\!]\theta))$$

Lemma D.0.39. conj_cons_disj_decide
For any clause c and propositional formulas f, g, if the disjunction $f \vee_p g$ is a logical consequence of c then f or g is a logical consequence of c

$$\forall c f g, \mathit{clause}(c) \to c \Rightarrow (f \vee_p g) \to ((c \Rightarrow f) \vee (c \Rightarrow g))$$

6.3.1

Lemma D.0.40. empty_relation_relates_bound
For any list of Boolean equations e, the empty relation relates only variables bound in e to variables bound in e.

$$\forall e, \emptyset \subseteq e \times e$$

Lemma D.0.41. reachable_exists_to
Given a relation R and propositional variables x, z. If z is reachable from x through R then there exists some variable y such that R relates y to z.

$$\forall R, \forall x\ z, R^+\ x\ z\ \to \exists y, R\ y\ z$$

Lemma D.0.42. reachable_exists_from
Given a relation R and propositional variables x, z. If z is reachable from x through R then there exists some variable y such that R relates x to y.

$$\forall R, \forall x\ z, R^{+}\ x\ z \rightarrow \exists y, R\ x\ y$$

Lemma D.0.43. conj_propForm_grows_right
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional formulas f and all clauses g_1, g_2, if the minimal environment consistent with R assigning *true* to all variables in g_1 assigns *true* to f, then the minimal environment consistent with R assigning *true* to all variables in g_1 and g_2 assigns *true* to f.

$$\begin{aligned}\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2,\\ \forall f g_1 g_2, \mathit{clause}(g_1) \rightarrow \mathit{clause}(g_2) \rightarrow [\![f]\!]\theta^{e_1,e_2,H}_{R,g_1} \rightarrow [\![f]\!]\theta^{e_1,e_2,H}_{R,(g_1 \wedge_p g_2)}\end{aligned}$$

Lemma D.0.44. conj_propForm_grows_left
Given some lists of Boolean equations e_1, e_2 and a relation $R \subseteq e_1 \times e_2$, then for all propositional formulas f and all clauses g_1, g_2, if the minimal environment consistent with R assigning *true* to all variables in g_2 assigns *true* to f, then the minimal environment consistent with R assigning *true* to all variables in g_1 and g_2 assigns *true* to f.

$$\begin{aligned}\forall R, \forall e_1 e_2, \forall H : R \subseteq e_1 \times e_2,\\ \forall f g_1 g_2, \mathit{clause}(g_1) \rightarrow \mathit{clause}(g_2) \rightarrow [\![f]\!]\theta^{e_1,e_2,H}_{R,g_2} \rightarrow [\![f]\!]\theta^{e_1,e_2,H}_{R,(g_1 \wedge_p g_2)}\end{aligned}$$

6.3.2

Lemma D.0.45. separe_subset_then_union_subset
For any relations R_1, R_2 which are subsets of lists of Boolean equations l_1, l_2, if both *is_subset* l_1 l_2 R_1 and *is_subset* l_1 l_2 R_2 hold then *is_subset* l_1 l_2 $(R_1 \cup R_2)$ also holds.

$$\begin{gathered}\forall l_1 l_2, \forall R_1 R_2, R_1 \subseteq l_1 \times l_2 \rightarrow R_2 \subseteq l_1 \times l_2 \rightarrow\\ l_1 \times l_2 \supseteq R_1 \rightarrow l_1 \times l_2 \supseteq R_2 \rightarrow l_1 \times l_2 \supseteq (R_1 \cup R_2)\end{gathered}$$

Lemma D.0.46. add_still_subset_left
For any relation R and lists of Boolean equations l_1, l_2 such that *is_subset* l_1 l_2 R, we can add any Boolean equation a to l_1, i.e. for any Boolean equation a we then have *is_subset* $(a :: l_1)$ l_2 R.

$$\forall l_2 l_1, \forall a, \forall R, l_1 \times l_2 \supseteq R \rightarrow (a :: l_1) \times l_2 \supseteq R$$

Lemma D.0.47. add_still_subset_right
For any relation R and lists of Boolean equations l_1, l_2 such that *is_subset* l_1 l_2

R, we can add any Boolean equation a to l_2, i.e. for any Boolean equation a we then have *is_subset* l_1 $(a :: l_2)$ R.

$$\forall l_2 l_1, \forall a, \forall R, l_1 \times l_2 \supseteq R \to l_1 \times (a :: l_2) \supseteq R$$

Lemma D.0.48. rev_card_gt0
For any relation R, lists of Boolean equations l_1, l_2 and natural number n, if *rev_subset_card* l_1 l_2 R n then $n > 0$.

$$\forall l_1 l_2, \forall \Gamma, \forall n, \#(l_1 \times l_2 \setminus \Gamma) = n \to n > 0$$

6.3.3

Lemma D.0.49. DNF_rel_cons_DNF_decidable
For any relation R relating only pairs of variables from $e_1 \times e_2$ and for any DNF's f, g, either g is a relative consequence of f under R or not.

$$\forall R, \forall e_1 e_2, R \subseteq (\mathit{flatten\_genBES}\ e_1) \times (\mathit{flatten\_genBES}\ e_2) \to \\ \forall f, \mathit{DNF}(f) \to \forall g, \mathit{DNF}(g) \to (f \overset{R}{\Rightarrow} g \vee \neg(f \overset{R}{\Rightarrow} g))$$

6.4

Lemma D.0.50. bnd_block_dec
For any list of Boolean equations l and propositional variable x, it is decidable if x is bound in l or not.

$$\forall l, \forall x, \{x \in l\} + \{x \notin l\}$$

Lemma D.0.51. bnd_dec
For any list of blocks e and propositional variable x, it is decidable if x is bound in e or not.

$$\forall e, \forall x, \{x \in e\} + \{x \notin e\}$$

6.4.1

Lemma D.0.52. rel_union_maintains_rel_cc
If two relations R_1, R_2 are consistent consequence relations on some BES E relative to some relation Γ, then the union of R_1 and R_2 is also a consistent consequence relation on E relative to Γ.

$$\forall E, \forall R_1 R_2 \Gamma, \mathit{relative\_cc}\ E\ R_1\ \Gamma \to \mathit{relative\_cc}\ E\ R_2\ \Gamma \to \\ \mathit{relative\_cc}\ E\ (R_1 \cup R_2)\ \Gamma$$

Lemma D.0.53. rel_cc_trans
Given a BES E and some relation on propositional variables Γ, relative consistent consequence on propositional formulas under E is transitive, i.e. for any propositional formulas a, b, c, if c is a consistent consequence of b relative to Γ, and b is a relative consistent consequence of a relative to Γ, then c is a relative consistent consequence of a relative to Γ.

$$\forall E, \forall \Gamma, \forall a\ b\ c, ((a \overset{\lessdot^E_\Gamma}{\Longrightarrow} b) \wedge (b \overset{\lessdot^E_\Gamma}{\Longrightarrow} c)) \rightarrow (a \overset{\lessdot^E_\Gamma}{\Longrightarrow} c)$$

Lemma D.0.54. bnd_block_flatten_bnd
Any variable bound in BES E is also bound in the result of flattening E to a list of Boolean equations.

$$\forall E, \forall x, x \in E \rightarrow x \in (\mathit{flatten\_genBES}\ E)$$

Lemma D.0.55. bnd_flatten_bnd_block
Any variable bound in the result of flattening BES E to a list of Boolean equations is also bound in E.

$$\forall E, \forall x, x \in (\mathit{flatten\_genBES}\ E) \rightarrow x \in E$$

6.5.1

Lemma D.0.56. AS1_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \wedge_p (b \wedge_p c))$ is a consistent consequence of $((a \wedge_p b) \wedge_p c)$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, (a \wedge_p (b \wedge_p c)) \overset{\lessdot^E_G}{\Longrightarrow} ((a \wedge_p b) \wedge_p c)$$

Lemma D.0.57. AS2_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $((a \wedge_p b) \wedge_p c)$ is a consistent consequence of $(a \wedge_p (b \wedge_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, ((a\ \ \wedge_p b) \wedge_p c) \overset{\lessdot^E_G}{\Longrightarrow} (a \wedge_p (b \wedge_p c))$$

Lemma D.0.58. AS3_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $((a \vee_p b) \vee_p c)$ is a consistent consequence of $(a \vee_p (b \vee_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, (a \vee_p (b \vee_p c)) \overset{\lessdot^E_G}{\Longrightarrow} ((a \vee_p b) \vee_p c)$$

Lemma D.0.59. AS4_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \vee_p (b \vee_p c))$ is a consistent consequence of $((a \vee_p b) \vee_p c)$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, ((a \vee_p b) \vee_p c) \overset{<^E_G}{\Longrightarrow} (a \vee_p (b \vee_p c))$$

Lemma D.0.60. COM1_sound
For any BES E, relation on propositional variables G and propositional formulas a, b. $(b \wedge_p a)$ is a consistent consequence of $(a \wedge_p b)$ on E, relative to G.

$$\forall E, \forall a\ b, \forall G, (a \wedge_p b) \overset{<^E_G}{\Longrightarrow} (b \wedge_p a)$$

Lemma D.0.61. COM2_sound
For any BES E, relation on propositional variables G and propositional formulas a, b. $(b \vee_p a)$ is a consistent consequence of $(a \vee_p b)$ on E, relative to G.

$$\forall E, \forall a\ b, \forall G, (a \vee_p b) \overset{<^E_G}{\Longrightarrow} (b \vee_p a)$$

Lemma D.0.62. DS1_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $((a \vee_p b) \wedge_p (a \vee_p c))$ is a consistent consequence of $(a \vee_p (b \wedge_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, (a \vee_p (b \wedge_p c)) \overset{<^E_G}{\Longrightarrow} ((a \vee_p b) \wedge_p (a \vee_p c))$$

Lemma D.0.63. DS2_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \vee_p (b \wedge_p c))$ is a consistent consequence of $((a \vee_p b) \wedge_p (a \vee_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, ((a \vee_p b) \wedge_p (a \vee_p c)) \overset{<^E_G}{\Longrightarrow} (a \vee_p (b \wedge_p c))$$

Lemma D.0.64. DS3_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $((a \wedge_p b) \vee_p (a \wedge_p c))$ is a consistent consequence of $(a \wedge_p (b \vee_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, (a \wedge_p (b \vee_p c)) \overset{<^E_G}{\Longrightarrow} ((a \wedge_p b) \vee_p (a \wedge_p c))$$

Lemma D.0.65. DS4_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \wedge_p (b \vee_p c))$ is a consistent consequence of $((a \wedge_p b) \vee_p (a \wedge_p c))$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, ((a \wedge_p b) \vee_p (a \wedge_p c)) \overset{<^E_G}{\Longrightarrow} (a \wedge_p (b \vee_p c))$$

Lemma D.0.66. AB1_sound
For any BES E, relation on propositional variables G and propositional formulas a, b. a is a consistent consequence of $(a \vee_p (a \wedge_p b))$ on E, relative to G.

$$\forall E, \forall a\ b, \forall G, (a \vee_p (a \wedge_p b)) \overset{<^{E}_{G}}{\Longrightarrow} a$$

Lemma D.0.67. AB2_sound
For any BES E, relation on propositional variables G and propositional formulas a, b. $(a \vee_p (a \wedge b))$ is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a\ b, \forall G, a \overset{<^{E}_{G}}{\Longrightarrow} (a \vee_p (a \wedge_p b))$$

Lemma D.0.68. ID1_sound
For any BES E, relation on propositional variables G and propositional formula a. $(a \wedge a)$ is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a, \forall G, a \overset{<^{E}_{G}}{\Longrightarrow} (a \wedge_p a)$$

Lemma D.0.69. ID2_sound
For any BES E, relation on propositional variables G and propositional formula a. a is a consistent consequence of $(a \vee_p a)$ on E, relative to G.

$$\forall E, \forall a, \forall G, (a \vee_p a) \overset{<^{E}_{G}}{\Longrightarrow} a$$

Lemma D.0.70. SUP_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \vee_p b)$ is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a\ b, \forall G, a \overset{<^{E}_{G}}{\Longrightarrow} (a \vee_p b)$$

Lemma D.0.71. INF_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. a is a consistent consequence of $(a \wedge b)$ on E, relative to G.

$$\forall E, \forall a\ b, \forall G, (a \wedge_p b) \overset{<^{E}_{G}}{\Longrightarrow} a$$

Lemma D.0.72. TOP_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. $(a \wedge \top)$ is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a, \forall G, a \overset{<^{E}_{G}}{\Longrightarrow} (a \wedge_p \top)$$

Lemma D.0.73. BOT_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. a is a consistent consequence of $(a \vee_p \bot)$ on E, relative to G.

$$\forall E, \forall a, \forall G, (a \vee_p \bot) \overset{<^E_G}{\Longrightarrow} a$$

Lemma D.0.74. CTX_sound
For any BES E, relation on propositional variables G, propositional formulas a, b, c and propositional variable x. If b is a consistent consequence of a on E, relative to G, then $c[x := b]$ is a consistent consequence of $c[x := a]$ on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall x, \forall G, (a \overset{<^E_G}{\Longrightarrow} b) \rightarrow ((c[x := a]) \overset{<^E_G}{\Longrightarrow} (c[x := b]))$$

Lemma D.0.75. TRA_sound
For any BES E, relation on propositional variables G and propositional formulas a, b, c. If b is a consistent consequence of a on E, relative to G and c is a consistent consequence of b on E, relative to G, then c is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a\ b\ c, \forall G, ((a \overset{<^E_G}{\Longrightarrow} b) \wedge (b \overset{<^E_G}{\Longrightarrow} c)) \rightarrow (a \overset{<^E_G}{\Longrightarrow} c)$$

Lemma D.0.76. REF_sound
For any BES E, relation on propositional variables G and propositional formula a. a is a consistent consequence of a on E, relative to G.

$$\forall E, \forall a, \forall G, (a \overset{<^E_G}{\Longrightarrow} a)$$

Lemma D.0.77. CC_sound
For any BES E, relation on propositional variables G, propositional formulas a, b, c and propositional variables x, y bound in E with equal rank. If the right hand side of x in E is a consistent consequence of the right hand side of y in E relative to $G \cup \{(x, y)\}$, then propositional formula y is a consistent consequence of propositional formula x on E, relative to G.

$$\forall E, \forall x\ y, \forall G, ((x \in E) \wedge (y \in E) \wedge (rank\ E\ x = rank\ E\ y)) \rightarrow$$
$$((rhs\ E\ x) \xrightarrow{<^E_{G \cup \{(x,y)\}}} (rhs\ E\ y)) \rightarrow ((var\ x) \overset{<^E_G}{\Longrightarrow} (var\ y))$$

Lemma D.0.78. CNT_sound
For any BES E, relation on propositional variables G and propositional variables $(x, y) \in G$. Propositional formula y is a consistent consequence of propositional formula x on E, relative to G.

$$\forall E, \forall x\ y, \forall G, ((x, y) \in G \rightarrow ((var\ x) \overset{<^E_G}{\Longrightarrow} (var\ y)))$$

6.5.2

Lemma D.0.79. conj_distribute_prv_tree
For any BES E, relation on propositional variables G clause f and DNF g, we can build a proof tree with $G \vdash (f \wedge_p g) \subset (dist(f)(g))$ as the root.

$$\forall E, \forall G, \forall f\ g, clause(f) \rightarrow DNF(g) \rightarrow prv\_tree\ E\ (G \vdash (f \wedge_p g) \subset (dist(f)(g)))$$

Lemma D.0.80. complete_to_bot
For any BES E, relation on propositional variables G and propositional formula f, if $\bot$ is a logical consequence of f then we can build a proof tree with $G \vdash f \subset \bot$ as the root.

$$\forall E, \forall G, \forall f, f \Rightarrow \bot \rightarrow prv\_tree\ E\ (G \vdash f \subset \bot)$$

Lemma D.0.81. complete_from_top
For any BES E, relation on propositional variables G and propositional formula f, if f is a logical consequence of $\top$ then we can build a proof tree with $G \vdash \top \subset f$ as the root.

$$\forall E, \forall G, \forall f, \top \Rightarrow f \rightarrow prv\_tree\ E\ (G \vdash \top \subset f)$$

Lemma D.0.82. complete_from_var
For any BES E, relation on propositional variables G, propositional formula f and propositional variable x, if f is a logical consequence of x then we can build a proof tree with $G \vdash x \subset f$ as the root.

$$\forall E, \forall G, \forall f, \forall x, (var\ x) \Rightarrow f \rightarrow prv\_tree\ E\ (G \vdash (var\ x) \subset f)$$

Lemma D.0.83. complete_to_var
For any BES E, relation on propositional variables G, propositional formula f and propositional variable x, if x is a logical consequence of f then we can build a proof tree with $G \vdash f \subset x$ as the root.

$$\forall E, \forall G, \forall f, \forall x, f \Rightarrow (var\ x) \rightarrow prv\_tree\ E\ (G \vdash f \subset (var\ x))$$

Lemma D.0.84. generalized_complete_from_conj_propForm
For any BES E, relation on propositional variables G, clause f and propositional formula g, if g is a logical consequence of f, then we can build a proof tree with $G \vdash f \subset g$ as the root.

$$\forall E, \forall G, \forall f\ g, clause(f) \rightarrow f \Rightarrow g \rightarrow prv\_tree\ E\ (G \vdash f \subset g)$$

6.5.3

Lemma D.0.85. complete_fom_conj_rel_cons
For any BES E, relation R on variables bound in E, relation G on propositional variables, clause f and DNF g such that g is a consequence of f relative to R, if we can make proof trees for all variables $(x, y) \in R$ with root $G \vdash x \subset y$, then we can create a proof tree with $G \vdash f \subset g$ as the root.

$$\begin{aligned}\forall E, \forall R\ G, R \subseteq (\mathit{flatten\_genBES}\ E) \times (\mathit{flatten\_genBES}\ E) \rightarrow \\ \forall f\ g, \mathit{clause}(f) \rightarrow \mathit{DNF}(g) \rightarrow f \overset{R}{\Rightarrow} g \rightarrow \\ (\forall x\ y, R\ x\ y \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y))) \rightarrow \\ \mathit{prv\_tree}\ E\ (G \vdash f \subset g)\end{aligned}$$

Lemma D.0.86. lem5_DNF
For any BES E, relation R on variables bound in E, relation G on propositional variables, DNF f and DNF g such that g is a consequence of f relative to R, if we can make proof trees for all variables $(x, y) \in R$ with root $G \vdash x \subset y$, then we can create a proof tree with $G \vdash f \subset g$ as the root.

$$\begin{aligned}\forall E, \forall R\ G, R \subseteq (\mathit{flatten\_genBES}\ E) \times (\mathit{flatten\_genBES}\ E) \rightarrow \\ \forall f\ g, \mathit{DNF}(f) \rightarrow \mathit{DNF}(g) \rightarrow f \overset{R}{\Rightarrow} g \rightarrow \\ (\forall x\ y, R\ x\ y \rightarrow \mathit{prv\_tree}\ E\ (G \vdash (\mathit{var}\ x) \subset (\mathit{var}\ y))) \rightarrow \\ \mathit{prv\_tree}\ E\ (G \vdash f \subset g)\end{aligned}$$

6.6

Lemma D.0.87. neq_no_rewrite
For any environment θ and propositional variables x, y such that $x \neq y$, redefining θ in y to any value, does not change the interpretation of x in θ.

$$\forall \theta, \forall x\ y, x \neq y \rightarrow \forall b, \theta(x) \leftrightarrow \theta[y := b](x)$$

Lemma D.0.88. unbound_no_change_block_function
For any list of Boolean equations *bl*, propositional variable x not bound in *bl* and environment θ, unfolding *bl* in θ does not change the interpretation of x in θ.

$$\forall bl, \forall x, x \notin bl \rightarrow \forall \theta, \theta(x) \leftrightarrow (||bl||\theta)(x)$$

Lemma D.0.89. b_propForm_solution_correct
The function *b_propForm_solution* is equivalent to *propForm_solution*.

$$\forall f, \forall \theta, [\![f]\!]\theta \leftrightarrow [\![f]\!]_b\theta$$

Lemma D.0.90. alt_tl
For any block h and list of blocks l, if the blocks in $h :: l$ have alternating fixedpoint symbols, then so do the blocks in l.

$$\forall l, \forall h, \mathit{alternating\_list\_block}\ (h :: l) \rightarrow \mathit{alternating\_list\_block}\ l$$

Lemma D.0.91. bnd_cnt_block_gt
For any list of Boolean equations *bl*, variables are bound in *bl* if and only if they occur at least once at the left hand side of an equation in *bl*.

$$\forall bl, \forall x, x \in bl \leftrightarrow \#(x \in bl) > 0$$

Lemma D.0.92. bnd_cnt_gt
For any list of blocks *bl*, variables are bound in *bl* if and only if they occur at least once at the left hand side of an equation in a block in *bl*.

$$\forall bl, \forall x, x \in bl \leftrightarrow \#(x \in bl) > 0$$

Lemma D.0.93. w_d_w_d_block
For any non-empty list of Boolean equations *bl*, if the BES consisting of a single block $\sigma\, bl$ (for $\sigma \in \{\mu, \nu\}$) is well formed, then *bl* is well formed.

$$\begin{gathered}\forall bl, \forall b, \forall \mathit{exists\_bl} : (\exists \mathit{beqn}, \mathit{In}\ \mathit{beqn}\ bl),\\ (\mathit{w\_d}\ ((\mathit{makeBlock}\ bl\ b\ \mathit{exists\_bl}) :: \mathit{nil})) \leftrightarrow \mathit{w\_d\_block}\ bl\end{gathered}$$

Lemma D.0.94. w_d_block_tl
For any non-empty list of Boolean equations *bl* and Boolean equation h, if h ::*bl* is well formed then so is *bl*.

$$\forall bl, \forall \mathit{heqn}, \mathit{w\_d\_block}\ (\mathit{heqn} :: bl) \rightarrow \mathit{w\_d\_block}\ bl$$

Lemma D.0.95. rhs_unfold_block
For any BES consisting of a single block *bl*, any variable x bound in this BES, and any θ, the interpretation of x under the unfolding of *bl* in θ is equivalent to the interpretation of the right hand side of x in *bl*.

$$\begin{gathered}\forall bl, \forall \mathit{alt\_bl} : \mathit{alternating\_list\_block}\ (bl :: \mathit{nil}),\\ \forall \mathit{w\_d\_bl} : \mathit{w\_d}\ (\mathit{make\_genBES}\ (bl :: \mathit{nil})\ \mathit{alt\_bl}), \forall x, x \in bl \rightarrow\\ \forall \theta, (||bl||\theta)(x) \leftrightarrow [\![\mathit{rhs}\ (\mathit{makeBES}\ (\mathit{make\_genBES}\ (bl :: \mathit{nil})\ \mathit{alt\_bl})\ \mathit{w\_d\_bl})\ x]\!]\theta\end{gathered}$$

Lemma D.0.96. w_d_tl
For any block *bh*, and any list of blocks *bl*, if *bh::bl* is well formed then so is *bl*.

$$\forall bl, \forall bh, \mathit{w\_d}\ (bh :: bl) \rightarrow \mathit{w\_d}\ bl$$

Lemma D.0.97. w_d_hd
For any block *bh*, and any list of blocks *bl*, if *bh::bl* is well formed then so is *bh*.

$$\forall bl, \forall bh, w\_d\ (bh :: bl) \to w\_d\_block\ bh$$

Lemma D.0.98. rank_tl
For any BES e, block *bl* such that *bl*:: e is well formed, and any variables x, y with equal rank in (*bl*:: e), the ranks of x and y in e are also equal.

$$\begin{aligned} &\forall bl, \forall e, \forall w\_d\_bl\_e : w\_d\ (bl :: e), \\ \forall x\ y, rank\ (bl :: e)\ x = rank\ (bl :: e)\ y \to rank\ e\ x = rank\ e\ y \end{aligned}$$

Lemma D.0.99. cc_tl
For any BES e, block *bl* such that *bl*:: e is a BES, relation R on propositional variables which is a consistent consequence relation on *bl*:: e, we have that R is also a consistent consequence relation on e.

$$\begin{aligned} \forall e, \forall bl, \forall alt\_bl\_e : alternating\_list\_block\ (bl :: e), \\ \forall w\_d\_bl\_e : w\_d\ (make\_genBES\ (bl :: e)\ alt\_bl\_e), \\ \forall R, cc\ (makeBES\ (make\_genBES\ (bl :: e)\ alt\_bl\_e)\ w\_d\_bl\_e)\ R \to \\ \forall alt\_e : alternating\_list\_block\ e, \forall w\_d\_e : w\_d\ (make\_genBES\ e\ alt\_e), \\ cc\ (makeBES\ (make\_genBES\ e\ alt\_e)\ w\_d\_e)\ R \end{aligned}$$

Lemma D.0.100. cc_hd
For any BES e, block *bl* such that *bl*:: e is a BES and *bl* on its own is also a BES, relation R on propositional variables which is a consistent consequence relation on *bl*:: e, we have that R is also a consistent consequence relation on *bl*.

$$\begin{aligned} \forall e, \forall bl, \forall alt\_bl\_e : alternating\_list\_block\ (bl :: e), \\ \forall w\_d\_bl\_e : w\_d\ (make\_genBES\ (bl :: e)\ alt\_bl\_e), \\ \forall R, cc\ (makeBES\ (make\_genBES\ (bl :: e) alt\_bl\_e)\ w\_d\_bl\_e)\ R \to \\ \forall alt\_bl : alternating\_list\_block\ (bl :: nil), \\ \forall w\_d\_bl : w\_d\ (make\_genBES\ (bl :: nil)\ alt\_bl), \\ cc\ (makeBES\ (make\_genBES\ (bl :: nil)\ alt\_bl)\ w\_d\_bl)\ R \end{aligned}$$

Bibliography

[1] Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013.

[2] H. Bekič. *Programming Languages and Their Definition*. 1984.

[3] G. Smolka C Doczkal. “Completeness and Decidability Results for CTL in Coq”. In: *ITP* (2014).

[4] Taolue Chen et al. “Equivalence Checking for Infinite Systems Using Parameterized Boolean Equation Systems”. In: *CONCUR*. Vol. 4703. Lecture Notes in Computer Science. Springer, 2007, pp. 120–135.

[5] Sjoerd Cranen et al. “Abstraction in Fixpoint Logic”. In: *ACM Trans. Comput. Log.* 16.4 (2015), p. 29.

[6] Sjoerd Cranen et al. “An Overview of the mCRL2 Toolset and Its Recent Advances”. In: *TACAS*. Vol. 7795. Lecture Notes in Computer Science. Springer, 2013, pp. 199–213.

[7] M. van Delft. *Consistent consequences in Coq*. 2016. URL: `https://github.com/MECvDelft/ConsequencesInCoq.git`.

[8] C. Paulin-Mohring G. Huet G. Kahn. *The Coq Proof Assistant*. 2016.

[9] Hubert Garavel et al. “CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes”. In: *CAV*. Vol. 4590. Lecture Notes in Computer Science. Springer, 2007, pp. 158–163.

[10] Maciej Gazda and Tim A. C. Willemse. “Consistent Consequence for Boolean Equation Systems”. In: *SOFSEM*. Vol. 7147. Lecture Notes in Computer Science. Springer, 2012, pp. 277–288.

[11] G. Gentzen. “Untersuchungen über das logische Schließen”. In: *Mathematische Zeitschrift* (1918).

[12] Jan Friso Groote and Radu Mateescu. “Verification of Temporal Properties of Processes in a Setting with Data”. In: *AMAST*. Vol. 1548. Lecture Notes in Computer Science. Springer, 1998, pp. 74–90.

[13] Jan Friso Groote and Tim A. C. Willemse. “Parameterised boolean equation systems”. In: *Theor. Comput. Sci.* 343.3 (2005), pp. 332–369.

[14] F. Kirchner H. Herbelin J. Narboux. *The Coq FAQ*. 2016. URL: `https://coq.inria.fr/faq`.

[15] “Homotopy Type Theory: Univalent Foundations of Mathematics”. In: *CoRR* abs/1308.0729 (2013).

[16] Pierre Letouzey. *Finite Set Library from Coq.* 2006. URL: `https://coq.inria.fr/library/Coq.Sets.Finite_sets.html`.

[17] P. Urzyczyn M. H. Sørensen. *Lectures on the Curry-Howard Isomorphism.* 1998.

[18] A. Mader. "Verification of Modal Properties Using Boolean Equation Systems". PhD thesis. Technische Universität München, 1997.

[19] Simona Orzan, Wieger Wesselink, and Tim A. C. Willemse. "Static Analysis Techniques for Parameterised Boolean Equation Systems". In: *TACAS.* Vol. 5505. Lecture Notes in Computer Science. Springer, 2009, pp. 230–245.

[20] Frank Pfenning and Christine Paulin-Mohring. "Inductively Defined Types in the Calculus of Constructions". In: *Mathematical Foundations of Programming Semantics.* Vol. 442. Lecture Notes in Computer Science. Springer, 1989, pp. 209–228.

[21] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction.* New York, NY, USA: Cambridge University Press, 2011. ISBN: 1107003636, 9781107003637.

[22] Alfred Tarski. "A lattice-theoretical fixpoint theorem and its applications." In: *Pacific J. Math.* 5.2 (1955), pp. 285–309.

[23] Tim A. C. Willemse. "Consistent Correlations for Parameterised Boolean Equation Systems with Applications in Correctness Proofs for Manipulations". In: *CONCUR.* Vol. 6269. Lecture Notes in Computer Science. Springer, 2010, pp. 584–598.

[24] Collective work. *The about Coq page.* 2016. URL: `https://coq.inria.fr/about-coq`.

[25] Dezhuang Zhang and Rance Cleaveland. "Fast Generic Model-Checking for Data-Based Systems". In: *FORTE.* Vol. 3731. Lecture Notes in Computer Science. Springer, 2005, pp. 83–97.