Consistent Consequence for PBES's

Martijn Struijs

June 3, 2018

1 Introduction

A Parametrised Boolean Equation Systems (PBES) can be used to describe certain model checking problems, so that the answer to the model checking problem can be obtained by solving the PBES. Unfortunately, this transformation often leads to a PBES that is too large to solve efficiently. One approach is to apply abstraction techniques that remove some information to reduce the size of the PBES, but retain enough information to be able to find a solution. The consistent consequence relation is a relation over the variables of a PBES that can be used to analyse the effect of abstraction techniques on these variables or even to directly find an abstraction.

However, proving whether two variables are related by consistent consequence can be quite complicated. A (sound and complete) derivation system is useful here, because this reduces the problem to finding a satisfying list of rules from the derivation system. This is simpler than proving consistent consequence 'from scratch' for proofs by hand. The derivation system could even be used to fully automate the process, since the system defines a much smaller solution space of possible proofs that might be effectively searched by a computer.

This report investigates how to extend the derivation system **CC** for the consistent consequence relation on the variables of a Boolean Equation Systems (BES), a special case of PBES's, from the paper by van Delft, Geuvers and Willemse [3] to the setting of PBES's. We create the system **PCC** for consistent consequence by replacing and adding rules to **CC** (see Table 1 for an overview). This system is sound relative to a derivation system for predicate logic. We don't know for what classes of PBES's the system **PCC** is complete, but can provide some ideas for future work in that direction.

The structure of the report is as follows. First, we introduce some prerequisites for describing PBES's and introduce the notion of consistent consequence formally. Then, we introduce the new rules for system **PCC** and illustrate how it works with some examples. After that, we show this system is relatively sound. Finally, we give some ideas and directions to investigate the completeness of the system.

2 Prerequisites, syntax and semantics

Before we can discuss a derivation system for consistent consequence in the context of PBES's, we must first be precise in what this means. Apart from some additional restrictions, the definition of consistent consequence is mostly the same as Definition 7.9[1] from Gazda's PhD thesis [1], although the presentation is more close to the start of section 4 of [3].

But first, we note the restrictions. We will only consider variables that are parameterised with at most one domain in our PBES's and only the domain $\mathbb{N}$, the natural numbers (of which the syntactic domain is Nat). The set Var contains all variables over the domain $\mathbb{N}$. For our PBES variables, we distinguish between *propositional variables*, called PVAR, and the *unary variables*, called UVAR. The propositional variables range over the two values in $\mathbb{B}$. The unary

[1] The difference is mostly notation and that the pairs from $\mathcal{X} \times \mathbb{N}$ here are called signatures there

variables associate a value in $\mathbb{B}$ with every element $\mathbb{N}$, i.e. they range over functions from $\mathbb{B}^{\mathbb{N}}$. We usually consider a propositional variable to be a unary variable of a constant function. To prevent possible confusion, we write $\mathcal{X} = \text{UVAR} \cup \text{PVAR}$, where the interpretation of the elements is as we just described. Additionally, we will not consider PBES's where there is universal or existential quantification on the variables from $\mathcal{X}$. We do allow this inside the boolean data terms, which cannot contain variables from $\mathcal{X}$.

Before we can give the definition of a consistent consequence relation, we first recall the formal definition of syntax and semantics of a PBES from Gazda [1, Section 7.2], tailored to the setting of this report:

2.1 Syntax

All expressions e ranging over the natural numbers that we will consider in the context of PBES's are of the form

$$e ::= e \text{ op } e \mid (e) \mid \underline{n} \mid x,$$

where $\underline{n} \in \text{Nat}$, $x \in \text{Var}$ and $\text{op} \in \{\oplus, \ominus, \otimes\}$. To simplify notation, we will often write the semantic version of an expression when no confusion over whether the expression should be interpreted syntactically or semantically can arise.

Definition 1 (PBES syntax)**.** *The syntax of all parametrised boolean equation system (PBES) $\mathcal{E}$ and predicate formulae ϕ, ψ considered in this report are as follows:*

$$\begin{aligned} \mathcal{E} &::= \emptyset \mid (\mu X(x : \text{Nat}) = \phi)\ \mathcal{E} \mid (\nu X(x : \text{Nat}) = \psi)\ \mathcal{E} \\ \phi, \psi &::= b \mid \phi \wedge \psi \mid \phi \vee \psi \mid X(e), \end{aligned}$$

where Nat *is the syntactic data domain of X (we only consider the naturals as data domain in this report), b is a boolean data term, $e \in E$ an expression over* Nat *and x a data variable of* Nat*. $\emptyset$ denotes the* empty *equation system. When an equation system is non-empty, we will omit the trailing $\emptyset$. In our setting, b can contain relations over* Nat *(such as* even*), quantifiers and other logical operators, interpreted as a boolean expression. A variable from $\mathcal{X}$ is called* bound *in some PBES $\mathcal{E}$ if it occurs on the left-hand side of a $=$ symbol in the PBES $\mathcal{E}$. The set of all bound variables in $\mathcal{E}$ is denoted by $\mathbf{bnd}_{\mathcal{E}}$.*

A simple example of a PBES is the following:

$$\mathcal{E}_1 : \qquad (\nu Z(n : \text{Nat}) = Z(n + 1)).$$

Note that $\mathbf{bnd}_{\mathcal{E}_1} = \{Z\}$. We will write f_X to denote the predicate formula on the right-hand side of the bound variable X in some equation system $\mathcal{E}$. So, in $\mathcal{E}_1$, $f_Z = Z(n + 1)$. We will assume that the PBES is 'locally closed', in the sense that all variables from Var are bound by either a quantifier (inside a boolean data term) or some predicate variable from $\mathcal{X}$ on the left-hand side of the equation in the PBES. For example, the variable n in $\mathcal{E}_1$ is bound by the predicate variable Z.

Furthermore, we will write $f(x)$ to indicate that x is the only free domain variable in formula f. If we have an expression e with one free variable, we can substitute e for x in f. We then write $f(e)$. Note that $f(e)$ has at most one free variable.

2.2 Semantics

To give semantics to the data, will use the *data environment* $\delta : \text{Var} \to \mathbb{N}$.

Notation 2. *For an environment δ and value v in the co-domain of δ, $\delta[x := v]$ denotes that δ is overrided by assigning value v to variable x: we have $\delta[x := v](y) = \delta(y)$ for all $y \neq x$, and $\delta[x := v](x) = v$. For a propositional formulae f and g, $f[X := g]$ denotes the substitution of all*

occurrences of the propositional variable X in the formula f by the formula g. For expressions e and d over Nat*, $e[x := d]$ denotes the substitution of the free variable x in e by the expression d.*

The semantics of the syntax for expressions is as follows, where additionally $e_1, e_2 \in E$ are arbitrary expressions and $n \in \mathbb{N}$ corresponds to $\underline{n} \in \mathrm{Nat}$:

$$\begin{aligned}
[\![x]\!]_\delta &= \delta(x) \\
[\![e_1 \oplus e_2]\!]_\delta &= [\![e_1]\!]_\delta + [\![e_2]\!]_\delta \\
[\![e_1 \ominus e_2]\!]_\delta &= [\![e_1]\!]_\delta - [\![e_2]\!]_\delta \\
[\![e_1 \otimes e_2]\!]_\delta &= [\![e_1]\!]_\delta \times [\![e_2]\!]_\delta \\
[\![\underline{n}]\!]_\delta &= n
\end{aligned}$$

For Boolean expressions b, we assume there is an interpretation such that $[\![b]\!]_\delta \in \mathbb{B}$.

To give semantics to a PBES, we also use an *environment* θ of a PBES: a function that assigns a function $f : \mathbb{N} \to \mathbb{B}$ to each variable $X \in \mathcal{X}$ in a PBES.

Definition 3 (predicate formula semantics)**.** *The semantics of a predicate formula ϕ in some PBES is defined relative to an environment $\theta : \mathcal{X} \to (\mathbb{N} \to \mathbb{B})$ and a data environment $\delta : \mathrm{Var} \to \mathbb{N}$:*

$$\begin{aligned}
[\![b]\!]_{\theta\delta} &= \begin{cases} \top & \textit{if } [\![b]\!]_\delta \textit{ holds} \\ \bot & \textit{otherwise} \end{cases} \\
[\![X(e)]\!]_{\theta\delta} &= \begin{cases} \top & \textit{if } \theta(X)[\![e]\!]_\delta \textit{ holds} \\ \bot & \textit{otherwise} \end{cases} \\
[\![\phi \wedge \psi]\!]_{\theta\delta} &= [\![\phi]\!]_{\theta\delta} \textit{ and } [\![\psi]\!]_{\theta\delta} \textit{ hold} \\
[\![\phi \vee \psi]\!]_{\theta\delta} &= [\![\phi]\!]_{\theta\delta} \textit{ or } [\![\psi]\!]_{\theta\delta} \textit{ hold}
\end{aligned}$$

We will use Δ to denote the set of all data environments. The solution of a PBES is the least or greatest fixpoint that is consistent with the equation in the system. To make this precise, first note that we can consider a predicate formula ϕ as a function that takes an $n \in \mathbb{N}$ and assigns it a truth value from $\mathbb{B}$. That is, it induces the function $\lambda v \in \mathbb{N}.[\![\phi]\!]_{\theta\delta[n:=v]}$, we which we denote by $[\![\phi_{\langle n \rangle}]\!]_{\theta\delta}$. Similarly, given environments θ, δ and a predicate variable $X : \mathbb{N} \to \mathbb{B}$, a boolean functional can be lifted to a predicate transformer T as follows:

$$T = \lambda f \in \mathbb{B}^{\mathbb{N}}.[\![\phi_{\langle n \rangle}]\!]_{\theta[X:=f]\delta}.$$

This transformer T has a least and greatest fixpoint over the boolean lattice. We denote its least fixpoint by μT and its greatest fixpoint by νT. The fixpoints μT and νT can be seen as the part of the solution of the PBES corresponding to variable X.

Definition 4 (PBES solution)**.** *The solution to a PBES $\mathcal{E}$ is an environment $\theta : \mathcal{X} \to (\mathbb{N} \to \mathbb{B})$, denoted by $[\![\mathcal{E}]\!]_{\theta\delta}$ The solution is defined recursively on $\mathcal{E}$:*

$$\begin{aligned}
[\![\emptyset]\!]_{\theta\delta} &:= \theta \\
[\![(\mu X(n : \mathrm{Nat}) = \phi)\ \mathcal{E}]\!]_{\theta\delta} &:= [\![\mathcal{E}]\!]_{\theta[X:=\mu T]\delta} \\
[\![(\nu X(n : \mathrm{Nat}) = \phi)\ \mathcal{E}]\!]_{\theta\delta} &:= [\![\mathcal{E}]\!]_{\theta[X:=\nu T]\delta},
\end{aligned}$$

where T is a defined above.

Finally, we recall the definition of a rank of a variable in $\mathbf{bnd}_\mathcal{E}$, the set of bound variables in PBES $\mathcal{E}$:

Definition 5 (Rank)**.** *Let $\mathcal{E}$ be a PBES. The* rank *of a predicate variable $X \in \mathbf{bnd}_{\mathcal{E}}$ is equal to the number of least and greatest fixpoint alternations that precede the equation that binds X, counting from 0 if the first equation is a greatest fixpoint and 1 otherwise.*

We say that two variables $X, Y \in \mathbf{bnd}_{\mathcal{E}}$ are in the same *block* if they have the same rank, denoted by $X \sim_{\mathcal{E}} Y$ or $X \sim Y$ when the equation system is clear from the context. We extend the relation $\sim_{\mathcal{E}}$ to all variables from $\mathcal{X}$ (i.e. also unbound variables) with the unique minimal extension, so $X \sim_{\mathcal{E}} Y$ for all $X, Y \in \mathcal{X} \setminus \mathbf{bnd}_{\mathcal{E}}$. In other words, two variables are in the same block if they occur in the same 'string' of identical fixpoints.

2.3 Consistent consequence

Now, we can start by reiterating the meaning of consistent consequence for this particular case:

A relation on predicate variables with a specific domain element induces a set of environments that are *consistent* with the relation:

Definition 6. *Let θ be an environment. We say that θ is* consistent *with a relation $R \subseteq (\mathcal{X} \times \mathbb{N})^2$ if for all $X, Y \in \mathcal{X}$ and $n, p \in \mathbb{N}$ such that $((X, n), (Y, p)) \in R$, we have $\theta(X)(n) \leq \theta(Y)(p)$. Here, $\leq$ is the relation on $\mathbb{B}^2$ such that $a \leq b$ when $a = b$ or $(a = \bot \wedge b = \top)$. The set of all environments consistent with R is denoted by Θ_R.*

Definition 7. *Let f, g be unary predicate formulae over* Nat *and let $n, p \in \mathbb{N}$. We say that (f, n) is a consequence of (g, p) relative to R, denoted $f(n) \overset{R}{\Rightarrow} g(p)$ if for all $\theta \in \Theta_R$ and all data environments δ we have $[\![f(x)]\!]\theta\delta[x := n] \leq [\![g(y)]\!]\theta\delta[y := p]$, where we write $e(x)$ to indicate that $x \in \mathrm{Var}$ is the only free variable in the formula e.*

Definition 8. *Let $\mathcal{E}$ be a PBES. A relation $R \subseteq (\mathcal{X} \times \mathbb{N})^2$ is a* consistent consequence *on $\mathcal{E}$ if, for all $((X, n), (Y, p)) \in R$, we have:*

1. *$X \sim_{\mathcal{E}} Y$ and;*
2. *If $X, Y \in \mathbf{bnd}_{\mathcal{E}}$ then also $f_X(n) \overset{R}{\Rightarrow} f_Y(p)$.*

We say that (Y, p) is a consistent consequence of (X, n), written $X(n) \lessdot_{\mathcal{E}} Y(p)$, if there exist a consistent consequence relation $R \subseteq (\mathbf{bnd}_{\mathcal{E}} \times \mathbb{N})^2$ such that $((X, n), (Y, p)) \in R$.

We will also need the notion of *relative* consistent consequences:

Definition 9. *Let $\Gamma \subseteq (\mathcal{X} \times \mathbb{N})^2$ be a relation on pairs of a predicate variable and a domain element. A relation $R \subseteq (\mathcal{X} \times \mathbb{N})^2$ is a* consistent consequence *on $\mathcal{E}$* relative to Γ *if, for all $((X, n), (Y, p)) \in R$, we have:*

1. *$X \sim_{\mathcal{E}} Y$; and*
2. *If $X, Y \in \mathbf{bnd}_{\mathcal{E}}$, then $f_X(n) \overset{R \cup \Gamma}{\Longrightarrow} f_Y(p)$.*

We say that $Y(p)$ is a consistent consequence of $X(n)$ relative to Γ, written $X(n) \lessdot_{\mathcal{E}}^{\Gamma} Y(p)$, if there exists a consistent consequence relation $R \subseteq (\mathbf{bnd}_{\mathcal{E}} \times \mathbb{N})^2$ on $\mathcal{E}$ relative to Γ such that $((X, n), (Y, p)) \in \Gamma \cup R$.

3 Rules for parameters

Consider again the PBES $\mathcal{E}_1$:

$$\mathcal{E}_1 : \qquad (\nu Z(n : \mathrm{Nat}) = Z(n + 1)).$$

Table 1: Rules for the derivation systems for consistent consequence **CC** and **PCC**, adapted from Table 1 in [3]. **CC** consists of the rules from blocks 1,2 and 3. **PCC** consists of the rules from blocks 1,2,4 and 5.

1. Axioms for negation-free propositional logic

rules of the form $\dfrac{}{\Gamma \vdash_{cc} A}$, where A ranges of the following laws:

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	$\alpha \vee \beta \subset \beta \vee \alpha$	AB1	$\alpha \subset \alpha \wedge (\alpha \vee \beta)$
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	$\alpha \vee \bot \subset \alpha$

2. Inequality logic rules

REF $\dfrac{}{\Gamma \vdash_{cc} \alpha \subset \alpha}$ TRA $\dfrac{\Gamma \vdash_{cc} \alpha \subset \beta \qquad \Gamma \vdash_{cc} \beta \subset \gamma}{\Gamma \vdash_{cc} \alpha \subset \gamma}$

CTX $\dfrac{\Gamma \vdash_{cc} \alpha \subset \beta}{\Gamma \vdash_{cc} \gamma[X := \alpha] \subset \gamma[X := \beta]}$ For some propositional formulae α, β

3. Consistent consequence rules

CC $\dfrac{\Gamma, X \subset Y \vdash_{cc} f_X \subset f_Y}{\Gamma \vdash_{cc} X \subset Y} X \sim Y$ CNT $\dfrac{}{\Gamma \vdash_{cc} X \subset Y} (X \subset Y) \in \Gamma$

4. Parametrised consistent consequence rules

PCC $\dfrac{\Gamma \cup \{(X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \mid \delta \in \Delta\} \vdash_{pcc} f_X(e) \subset f_Y(e')}{\Gamma \vdash_{pcc} X(e) \subset Y(e')} X \sim Y$

PCNT $\dfrac{}{\Gamma \vdash_{pcc} X(e) \subset Y(e')} (X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \in \Gamma$ for all $\delta \in \Delta$

5. Parameter logic rules

GEN $\dfrac{\Gamma \vdash_{pcc} X(e) \subset Y(e')}{\Gamma \vdash_{pcc} X(e[x := d]) \subset Y(e'[x := d])}$ For some expression d

DLOGIC $\dfrac{\Vdash f \Rightarrow f'}{\Gamma \vdash_{pcc} f \subset f'}$

If we interpret a variable paired with an element from Nat, such as $(Z, 1)$, as a propositional variable 'Z_1', then we can apply rules from **CC** to the variables in this system.

Suppose we want to derive $\emptyset \vdash_{cc} Z(2) \subset Z(1)$. We cannot gain any progress using the rules that ignore $f_Z(n)$. So, the only rule that can be applied is CC. But if we apply that rule to try to find a relation with $Z(k)$ for some $k \in$ Nat, we must derive something involving $f_Z(k) = Z(k+1)$, of which we can again only try to make progress by applying CC thereby get a derivation tree without end. So, the rules from **CC** cannot be used to derive consistent consequence for the variables of $\mathcal{E}_1$, while we have that $Z(k) \ll Z(l)$ for all $k, l \in$ Nat.

So, the derivation system for BES, 'lifted' to the PBES case, is not complete. Therefore, we change and add some rules to get the system **PCC**, with which we can at least derive all true consistent consequence relations for the examples in this report. An overview of the rules for both systems can be found in Table 1.

3.1 New rules

In a sense, parameters will allow us to make statements about a potentially infinite number of domain elements simultaneously, which gives our derivation system more power. Let $X, Y \in \mathcal{X}$, e, e' expressions over $\mathbb{N}$, f_X, f_Y the RHS of the variables X, Y in their PBES $\mathcal{E}$, respectively and Δ the set of all data environments. So, we define the rule:

$$\textsf{PCC}\ \frac{\Gamma \cup \{(X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \mid \delta \in \Delta\} \vdash_{pcc} f_X(e) \subset f_Y(e')}{\Gamma \vdash_{pcc} X(e) \subset Y(e')}\ X \sim Y$$

The context rule also needs to be adapted to take data into account.

$$\textsf{PCNT}\ \frac{}{\Gamma \vdash_{pcc} X(e) \subset Y(e')}\ (X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \in \Gamma \text{ for } \textit{all } \delta \in \Delta$$

The first 'really new' rule concerns generalisation of expressions in the domain (potentially using parameters):

$$\textsf{GEN}\ \frac{\Gamma \vdash_{pcc} X(e) \subset Y(e')}{\Gamma \vdash_{pcc} X(e[x := d]) \subset Y(e'[x := d])}\ \text{For some expression } d$$

Note that the rule GEN may appear a bit weak, as we can only generalise on variables, not on formulas. However, if all we want to conclude is $X(n) \subset Y(m)$ for concrete $m, n \in \mathbb{N}$ by generalising at some point, we would likely have to be able derive $X(e) \subset Y(e')$ for some expressions e, e' which 'generalise n, m' at some point. Hence, this rule may actually be powerful enough.

The following rules is an 'interface' to the part of our derivation system on logic over expressions or negation free first order logic. $\Vdash$ denotes a judgement from first order logic and we give the following rule:

$$\textsf{DLOGIC}\ \frac{\Vdash f \Rightarrow f'}{\Gamma \vdash_{pcc} f \subset f'}$$

This rule states that all implications over first order negation free logic formulas are a consistent consequence. A consequence of this rule is that the system **PCC** can at most be sound or complete relative to the soundness or completeness of first order negation free logic. While this is the simplest form of this rule, we often apply this rule to get the following rules:

$$\textsf{SLOGIC}\ \frac{\Vdash f \Rightarrow f' \qquad \Gamma \vdash_{pcc} f' \subset g' \qquad \Vdash g' \Rightarrow g}{\Gamma \vdash_{pcc} f \subset g}$$

$$\mathsf{MLOGIC}\ \frac{\Vdash f \Rightarrow f' \qquad \Gamma \vdash_{pcc} g \subset g'}{\Gamma \vdash_{pcc} g[X := f] \subset g'[X := f']}$$

Rule MLOGIC follows from DLOGIC and CTX:

$$\dfrac{\dfrac{\dfrac{\Vdash f \Rightarrow f'}{\Gamma \vdash_{pcc} f \subset f'}\,\mathsf{DLOGIC}}{\Gamma \vdash_{pcc} g[X := f] \subset g[X := f']}\,\mathsf{CTX} \qquad \dfrac{\Gamma \vdash_{pcc} g \subset g'}{\Gamma \vdash_{pcc} g[X := f'] \subset g'[X := f']}\,\text{Lemma 10}}{\Gamma \vdash_{pcc} g[X := f] \subset g'[X := f']}\,\mathsf{TRA}$$

Where we use the following Lemma:

Lemma 10. *If* $\Gamma \vdash_{pcc} g \subset g'$*, then* $\Gamma \vdash_{pcc} g[X := f] \subset g'[X := f]$ *for any formula* f*.*

Note that while this Lemma holds, we cannot actually make a derivation tree to derive $g[X := f] \subset g[X := f]$ from $g \subset g'$.

Rule SLOGIC follows from DLOGIC and TRA:

$$\dfrac{\dfrac{\Vdash f \Rightarrow f'}{\Gamma \vdash_{pcc} f \subset f'}\,\mathsf{DLOGIC} \qquad \dfrac{\Gamma \vdash_{pcc} f' \subset g' \qquad \dfrac{\Vdash g' \Rightarrow g}{\Gamma \vdash_{pcc} g' \subset g}\,\mathsf{DLOGIC}}{\Gamma \vdash_{pcc} f' \subset g}\,\mathsf{TRA}}{\Gamma \vdash_{pcc} f \subset g}\,\mathsf{TRA}$$

3.2 Some examples

Note that in the examples in this section and the rest of the report, we will omit the application of the rules for commutativity for the sake of brevity.

We can now derive consistent consequence on $\mathcal{E}_1$. Note that for any $k, l \in \mathbb{N}$ that $Z(k) \lessdot Z(l)$. For simplicity, we show how to derive $Z(1) \subset Z(0)$ in this example:

$$\dfrac{\dfrac{\dfrac{}{\{(Z, [\![k]\!]_\delta) \subset (Z, [\![l]\!]_\delta) \mid \delta \in \Delta\} \vdash_{pcc} Z(k+1) \subset Z(l+1)}\,\mathsf{PCNT}}{\emptyset \vdash_{pcc} Z(k) \subset Z(l)}\,\mathsf{PCC}}{\emptyset \vdash_{pcc} Z(1) \subset Z(0)}\,\mathsf{GEN}$$

A slightly more interesting PBES is the following:

$$\mathcal{E}_2 : \qquad (\nu Z(n : \mathrm{Nat}) = Z(n+2) \wedge \mathrm{even}(n)).$$

Here, we have $Z(k) \lessdot Z(l)$ for all $k, l \in \mathbb{N}$ such that $\mathrm{even}(l) \Rightarrow \mathrm{even}(k)$. However, we do *not* have $Z(k) \lessdot Z(l)$ for k even and l odd. This means that we cannot derive $\emptyset \vdash_{pcc} Z(k) \subset Z(l)$, as it would be false for k even and l odd (and our system is sound)[2]. So, we will make a separate derivation for the three truth values of the pair $(\mathrm{even}(k), \mathrm{even}(l))$ for which $\emptyset \vdash_{pcc} Z(k) \lessdot Z(l)$ holds.

First, we derive $Z(2) \subset Z(0)$, where $\Gamma := \{(Z, [\![2k]\!]_\delta) \subset (Z, [\![2n]\!]_\delta) \mid \delta \in \Delta\}$,

$$\dfrac{\dfrac{\dfrac{\Vdash \mathrm{even}(2k) \Rightarrow \mathrm{even}(2n) \qquad \dfrac{\dfrac{}{\Gamma \vdash_{pcc} Z(2k+2) \subset Z(2n+2)}\,\mathsf{PCNT}}{\Gamma \vdash_{pcc} Z(2k+2) \wedge X \subset Z(2n+2) \wedge X}\,\mathsf{CTX}}{\Gamma \vdash_{pcc} Z(2k+2) \wedge \mathrm{even}(2k) \subset Z(2n+2) \wedge \mathrm{even}(2n)}\,\mathsf{MLOGIC}}{\emptyset \vdash_{pcc} Z(2k) \subset Z(2n)}\,\mathsf{PCC}}{\emptyset \vdash_{pcc} Z(2) \subset Z(0)}\,\mathsf{GEN}$$

[2]Note that for example the statement '$\emptyset \vdash_{pcc} Z(k) \subset Z(l)$ for $\mathrm{even}(k) \wedge \mathrm{even}(n)$' can be equivalently expressed as '$\emptyset \vdash_{pcc} Z(2k) \subset Z(2l)$'. But there doesn't seem to be a *single* expression of the form $\emptyset \vdash_{pcc} Z(e) \subset Z(e')$ for the statement '$\emptyset \vdash_{pcc} Z(k) \subset Z(l)$ for $\mathrm{even}(k) \Rightarrow \mathrm{even}(n)$'.

Next, we derive $Z(3) \subset Z(1)$, where $\Gamma := \{(Z, [\![3]\!]_\delta) \subset (Z, [\![1]\!]_\delta) \mid \delta \in \Delta\}$:

$$\dfrac{\dfrac{\Vdash Z(5) \wedge \mathrm{even}(3) \Rightarrow \bot \qquad \overline{\Gamma \vdash_{pcc} \bot \subset \bot} \qquad \Vdash \bot \Rightarrow Z(3) \wedge \mathrm{even}(1)}{\Gamma \vdash_{pcc} Z(5) \wedge \mathrm{even}(3) \subset Z(3) \wedge \mathrm{even}(1)}\,\mathsf{SLOGIC}}{\emptyset \vdash_{pcc} Z(3) \subset Z(1)}\,\mathsf{PCC}$$

I will omit showing why $\Gamma \vdash_{pcc} \bot \subset F$ holds for all Γ and F, as this is easy to derive using the negation free propositional logic axioms. Note that this case is rather trivial, as we didn't even need the generalisation rule.

Finally, we derive for $Z(1) \subset Z(0)$:

$$\dfrac{\dfrac{\Vdash Z(3) \wedge \mathrm{even}(1) \Rightarrow \bot \qquad \overline{\Gamma \vdash_{pcc} \bot \subset Z(2)} \qquad \Vdash Z(2) \Rightarrow Z(2) \wedge \mathrm{even}(0)}{\Gamma \vdash_{pcc} Z(3) \wedge \mathrm{even}(1) \subset Z(2) \wedge \mathrm{even}(0)}\,\mathsf{LOGIC}}{\emptyset \vdash_{pcc} Z(1) \subset Z(0)}\,\mathsf{PCC}$$

Again, no generalisation is needed.

What happens when we try to derive something we'd not like to be derivable, such as $Z(0) \subset Z(1)$?

$$\dfrac{\dfrac{\dfrac{???}{\Gamma \vdash_{pcc} Z(n+2) \wedge \mathrm{even}(n) \subset Z(n+3) \wedge \mathrm{even}(n+1)}\,\mathsf{PCC}}{\emptyset \vdash_{pcc} Z(n) \subset Z(n+1)}\,\mathsf{GEN}}{\emptyset \vdash_{pcc} Z(0) \subset Z(1)}$$

First note that the GEN step all we can do at first, as CC (without parameters) is sound. Applying some form of PCC is again all we can do next, as we want to make a claim over PBES variables. But then we are stuck, because DLOGIC cannot derive $\mathrm{even}(n+1)$, as this derivation system doesn't know whether n is even or not! (this is the reason why we generalised to $2n$ in the previous cases. Of course, if we would do that here we would have to derive $Z(0) \subset \bot$, which is nonsense that hopefully is impossible to derive.)

Next, we consider a simple example with more than one variable:

$$\mathcal{E}_3 : \qquad \nu\langle (X(n : \mathrm{Nat}) = Y(n) \vee X(n+1)) \quad (Y(n : \mathrm{Nat}) = X(n+2)) \rangle.$$

Here, we show how to derive $X(2) \subset Y(1)$, with $\Gamma = \{(X, [\![n+1]\!]_\delta) \subset (Y, [\![n]\!]_\delta) \mid \delta \in \Delta\}$, $\Gamma' = \{(Y, [\![n+1]\!]_\delta) \subset (X, [\![n+2]\!]_\delta) \mid \delta \in \Delta\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{\Gamma \cup \Gamma' \vdash_{pcc} X(n+3) \subset Y(n+2) \vee X(n+3)}\,\mathsf{SUP}}{\Gamma \vdash_{pcc} Y(n+1) \subset X(n+2)}\,\mathsf{PCC}}{\Gamma \vdash_{pcc} Y(n+1) \vee X(n+2) \subset X(n+2) \vee X(n+2)}\,\mathsf{CTX} \qquad \dfrac{}{\Gamma \vdash_{pcc} X(n+2) \vee X(n+2) \subset X(n+2)}\,\mathsf{ID2}}{\Gamma \vdash_{pcc} Y(n+1) \vee X(n+2) \subset X(n+2)}\,\mathsf{TRA}}{\emptyset \vdash_{pcc} X(n+1) \subset Y(n)}\,\mathsf{PCC}}{\emptyset \vdash_{pcc} X(2) \subset Y(1)}\,\mathsf{GEN}$$

4 Soundness

We call the new derivation system **PCC** *sound* if for all valid judgements of $\emptyset \vdash_{pcc} f(e) \subset g(e')$, we have $f([\![e]\!]_\delta) \preccurlyeq_{\mathcal{E}} g([\![e']\!]_\delta)$ for all environments δ. To prove soundness, it is useful to extend this notion to judgements and to single rules:

Definition 11. *A judgement* $\Gamma \vdash_{pcc} f(e) \subset g(e')$ *is called* sound *if for all data environments* δ*, we have* $f([\![e]\!]_\delta) \preccurlyeq^\Gamma_{\mathcal{E}} g([\![e']\!]_\delta)$.

A derivation rule R *is called* sound *if, whenever all premises of the rule are sound, the conclusion of the rule is sound. (both the premisses and conclusions are (sets of) judgements)*

Note that we have chosen our definitions such that the derivation system is sound if all rules are sound:

Lemma 12. *The derivation system* ***PCC*** *is sound if all rules are sound.*

Proof. Suppose we have a derivation tree $\mathcal{T}$ of PCC that derives $\Gamma \vdash_{pcc} f(e) \subset g(e')$. We claim that this judgement is then sound. To prove this claim, we use induction on the length of the derivation tree. If the length is 0, the derivation is trivially valid, so the base case holds. For the inductive step, suppose the claim holds for all derivation trees of length at most n, for some $n \in \mathbb{N}$. If the length of the tree is $n+1$, then consider the final rule that is applied. All premises of this rule are derived with a derivation tree of length at most n, so the premises are sound. Since the rule is sound, the conclusion is sound and therefore the final derivation is sound. By induction, our claim follows.

So, if we take $\Gamma = \emptyset$, our claim is that if we have a derivation tree for $\emptyset \vdash_{pcc} f(e) \subset g(e')$, we have that $f([\![e]\!]_\delta) \preccurlyeq_{\mathcal{E}} g([\![e']\!]_\delta)$. □

So, we will prove the soundness of our derivation system **PCC** by proving that each individual rule is sound. We first prove correctness of the most important rule, PCC.

Lemma 13. *The rule PCC is sound.*

Proof. Suppose the premiss $\Gamma' \vdash_{pcc} f_X(e) \subset f_Y(e')$ is sound, where $\Gamma' = \Gamma \cup \{(X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \mid \delta \in \Delta\}$ with $X \sim_{\mathcal{E}} Y$. This means that, for all δ, we have $f_X([\![e]\!]_\delta) \overset{\preccurlyeq^{\Gamma'}_{\mathcal{E}}}{\Longrightarrow} f_Y([\![e']\!]_\delta)$. So, by the definition of relative consequence, there exists a relation $R' \subseteq (\mathbf{bnd}_{\mathcal{E}} \times \mathbb{N})^2$ such that $f_X([\![e]\!]_\delta) \overset{\Gamma' \cup R'}{\Longrightarrow} f_Y([\![e']\!]_\delta)$. Let R' be such and define $R := R' \cup \{(X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta) \mid \delta \in \Delta\}$.

Note that R is a consistent consequence relation relative to Γ: For any $((W, n) \subset (Z, p)) \in R$, we have $W \sim_{\mathcal{E}} Z$, as either W and Z are part of the relation R', or we have $W = X$ and $Z = Y$ for which $X \sim_{\mathcal{E}} Y$ is given. Furthermore, if $W, Z \in \mathbf{bnd}_{\mathcal{E}}$, then $f_W(n) \overset{\Gamma \cup R}{\Longrightarrow} f_Z(p)$, since $R' \cup \Gamma' = R \cup \Gamma$. So R is indeed a consistent consequence relation relative to Γ.

As $((X, [\![e]\!]_\delta) \subset (Y, [\![e']\!]_\delta)) \in R \cup \Gamma$, we have that $X([\![e]\!]_\delta) \preccurlyeq^\Gamma_{\mathcal{E}} Y([\![e']\!]_\delta)$. So, the conclusion of rule PCC, $\Gamma \vdash_{pcc} X(e) \subset Y(e')$, is sound. □

Lemma 14. *The rule PCNT is sound.*

Proof. To validly apply the rule, we require that $(X, [\![e]\!]_\delta), (Y, [\![e']\!]_\delta) \in \Gamma$ for all δ. Note that the relation $R = \emptyset$ is trivially a consistent consequence relation relative to Γ. As $(X, [\![e]\!]_\delta), (Y, [\![e']\!]_\delta) \in \Gamma \cup R$, we have $X([\![e]\!]_\delta) \preccurlyeq^\Gamma_{\mathcal{E}} X([\![e']\!]_\delta)$ for all δ. So, the conclusion of the rule, $\Gamma \vdash_{pcc} X(e) \subset Y(e')$, is sound. □

Lemma 15. *The rule GEN is sound.*

Proof. First, note that if $\delta(x) = [\![d]\!]_\delta$, then $[\![e]\!]_\delta = [\![e[x := d]]\!]_\delta$. This means that $\{[\![e[x := d]]\!]_\delta \mid \delta \in \Delta\} = \{[\![e]\!]_\delta \mid \delta \in \Delta : \delta(x) = [\![d]\!]_\delta\} \subseteq \{[\![e]\!]_\delta \mid \delta \in \Delta\}$. Suppose the premiss of the rule, $\Gamma \vdash_{pcc} X(e) \subset Y(e')$, is sound. Then, we have that $X([\![e]\!]_\delta) \preccurlyeq^\Gamma_{\mathcal{E}} Y([\![e']\!]_\delta)$ for all δ. By the set inclusion above, it follows that $X([\![e[x := d]]\!]_\delta) \preccurlyeq^\Gamma_{\mathcal{E}} Y([\![e'[x := d]]\!]_\delta)$ for all δ. So, the conclusion of the rule is sound. □

Lemma 16. *The rule DLOGIC is sound, relative to the soundness of* $\Vdash$.

Proof. This follows directly from the definition of DLOGIC. □

So, all new rules but DLOGIC are sound and the old rules (that are not replaced, so except CC and CNT) for consistent consequence on BES remain sound for PBES case, as those are independent of the equation system. Since a sound rule is also relatively sound to $\Vdash$, we have that all rules are sound relative to $\Vdash$. So, we get the theorem

Theorem 17. *The derivation system **PCC** is sound relative to the soundness of* $\Vdash$.

5 Completeness

A derivation system is *complete* if all true statements can be derived using the system, i.e. there is a finite derivation tree ending in $\emptyset \vdash_{pcc} X(e) \subset Y(e')$ for all X, Y, e, e' such that $X(\llbracket e \rrbracket_\delta) \ll Y(\llbracket e \rrbracket_\delta)$ for all δ.

The derivation system CC has been shown to be complete for consistent consequence on Boolean Equation Systems [3]. The strategy from that paper is to first decompose all formulas to disjunctive normal form and then note that applying the CC rule sufficiently often is sufficient to derive the statement. Given that in the BES setting, $|\Gamma| \leq |\mathbf{bnd}_{\mathcal{E}}| < \infty$ for all contexts Γ, the rule CC only has to be applied a finite amount of times, so we get a finite derivation tree as a result.

5.1 Completeness for cases similar to BES

The basic idea from the BES case is applicable to system **PCC**. However, Γ is usually infinite after even the first application of rule PCC. An argument to bound the number of applications of the rule PCC would likely be sufficient to prove completeness. Having such a bound is clearly necessary, so it seems that this crucial part that should be investigated to determine the completeness of this system.

Note that while the size of Γ cannot be bounded, there are cases where the number of applications of PCC that actually adds relations to Γ is finite, for a well chosen generalisation for GEN. Consider equation system $\mathcal{E}_3$, for example. If we want to derive $\emptyset \vdash_{pcc} X(n) \subset Y(k)$, then we can do so by applying of PCC at most 4 times, once for every ordered pair of X, Y. We can do this by using GEN to introduce new variables and then apply PCC, such that we always add a set of the form $\{(Z, x) \subset (Z', y) \mid x, y \in \mathbb{N}\}$ with $Z, Z' \in \{X, Y\}$. This means that after applying rule PCC for all four ordered pairs, Γ contains the largest possible relation, from which we are able to derive any consistent consequence.

This technique works because there are no boolean data terms that can only be evaluated properly for a 'limited' generalisation – such as even(k), which can only be reduced to a boolean value if $k = 2e$ or $k = 2e + 1$ for some expression e – and therefore we always can reduce the intermediate judgements in the derivation to a number of relations of the form $\Gamma \vdash_{pcc} Z(k) \subset Z'(l)$. So, for this case without boolean data terms, it seems likely that we can still bound the number of applications of PCC. Do note that this case is rather 'simple', in the sense that 'merely' static analysis, such as in [2, Section 4], is already enough to determine the solution of these equation systems. Do note that proving the correctness of these methods can be seen as a form of finding a consistent consequence (by hand). Of course, we argue that this case is essentially the same as the BES case, so it isn't surprising that solving this case is 'simple'.

For a concrete example of this technique, consider the following equation system:

$$\mathcal{E}_4 : \qquad (\nu X(n : \mathrm{Nat}) = Y(n+2) \vee X(n \times n)) \quad (\nu Y(n : \mathrm{Nat}) = X(n+1)).$$

First, note that we can derive $\emptyset \vdash_{pcc} Y(1) \subset X(2)$. Here,

$$\begin{aligned}
\Gamma_1 &= \{(Y, \llbracket n_1 \rrbracket_\delta) \subset (X, \llbracket k_1 \rrbracket_\delta) \mid \delta \in \Delta\} \\
\Gamma_2 &= \{(X, \llbracket n_2 \rrbracket_\delta) \subset (X, \llbracket k_2 \rrbracket_\delta) \mid \delta \in \Delta\} \\
\Gamma_3 &= \{(X, \llbracket n_3 \rrbracket_\delta) \subset (Y, \llbracket k_3 \rrbracket_\delta) \mid \delta \in \Delta\},
\end{aligned}$$

and $\Gamma_{1,\dots,i} := \Gamma_1 \cup \ldots \cup \Gamma_i$

$$\dfrac{\dfrac{}{\Gamma_{1,2,3} \vdash_{pcc} Y(n_3+2) \vee Y(n_3+2) \subset Y(n_3+2)}\,\mathsf{ID2} \quad \dfrac{}{\Gamma_{1,2,3} \vdash_{pcc} Y(n_3+2) \subset X(k_3+1)}\,\mathsf{PCNT}}{\Gamma_{1,2,3} \vdash_{pcc} Y(n_3+2) \vee Y(n_3+2) \subset X(k_3+1)}\,\mathsf{TRA}$$

A

$$\dfrac{\dfrac{\dfrac{}{\Gamma_{1,2,3} \vdash_{pcc} X(n_3^2) \subset Y(n_3+2)}\,\mathsf{PCNT}}{\Gamma_{1,2,3} \vdash_{pcc} Y(n_3+2) \vee X(n_3^2) \subset Y(n_3+2) \vee Y(n_3+2)}\,\mathsf{CTX} \quad \text{A}}{\dfrac{\Gamma_{1,2,3} \vdash_{pcc} Y(n_3+2) \vee X(n_3^2) \subset X(k_3+1)}{\dfrac{\Gamma_{1,2} \vdash_{pcc} X(n_3) \subset Y(k_3)}{\dfrac{\Gamma_{1,2} \vdash_{pcc} X(n_2^2) \subset Y(k_2+2)}{\Gamma_{1,2} \vdash_{pcc} Y(n_2+2) \vee X(n_2^2) \subset Y(n_2+2) \vee Y(k_2+2)}\,\mathsf{CTX}}\,\mathsf{GEN}}\,\mathsf{PCC}}\,\mathsf{TRA}$$

B

$$\dfrac{\dfrac{}{\Gamma_{1,2} \vdash_{pcc} Y(n_2+2) \subset X(k_2^2)}\,\mathsf{PCNT}}{\Gamma_{1,2} \vdash_{pcc} Y(n_2+2) \vee Y(k_2+2) \subset Y(k_2+2) \vee X(k_2^2)}\,\mathsf{CTX}$$

C

$$\dfrac{\dfrac{\dfrac{\text{B} \qquad \text{C}}{\Gamma_{1,2} \vdash_{pcc} Y(n_2+2) \vee X(n_2^2) \subset Y(k_2+2) \vee X(k_2^2)}\,\mathsf{TRA}}{\Gamma_1 \vdash_{pcc} X(n_2) \subset X(k_2)}\,\mathsf{PCC}}{\dfrac{\Gamma_1 \vdash_{pcc} X(n_1+1) \subset X(k_1^2)}{\dfrac{\Gamma_1 \vdash_{pcc} X(n_1+1) \subset Y(k_1+2) \vee X(k_1^2)}{\dfrac{\emptyset \vdash_{pcc} Y(n_1) \subset X(k_1)}{\emptyset \vdash_{pcc} Y(1) \subset X(2)}\,\mathsf{GEN}}\,\mathsf{PCC}}\,\mathsf{TRA}} \quad \dfrac{}{\Gamma_1 \vdash_{pcc} X(k_1^2) \subset Y(k_1+2) \vee X(k_1^2)}\,\mathsf{SUP}$$

To conclude, as long as there are no boolean data terms in an equation system, the similarity to a BES can likely be used to show completeness for these systems.

5.2 Completeness with boolean data terms

For the next examples, we add an additional 'rounded division' operator for the data expressions: define $\div : \mathbb{N} \times \mathbb{N}_+ \to \mathbb{N}$, with $n \div k = \lceil \frac{n}{k} \rceil$ for all $n \in \mathbb{N}, k \in \mathbb{N}_+$. We also write $A \Rightarrow B$ as a shorthand for $\neg A \vee B$. Consider the following equation system:

$$\mathcal{E}_5 = (\mu Z(n : \text{Nat}) = [\text{even}(n) \Rightarrow Z(n \div 2)] \wedge [\neg\,\text{even}(n) \Rightarrow Z((3n+1) \div 2)]).$$

As we have $\text{even}(n) \vee \neg\,\text{even}(n) = \top$ for all n, the value of variable Z always depends on itself. So, we get that $Z(i) \ll Z(j)$ for all $i, j \in \mathbb{N}$. Suppose we want to derive $\emptyset \vdash_{pcc} Z(2) \subset Z(3)$.

It seems natural to first try to make the generalisation to $\emptyset \vdash_{pcc} Z(2n) \subset Z(2k+1)$, as we can use the rule DLOGIC to replace the $\text{even}(\bullet)$ conditions by booleans and simplify the formulas to the form. However, after applying PCC, we then have to derive $\Gamma \vdash_{pcc} Z(n) \subset Z(3k+2)$, with $\Gamma = \{(Z, [\![2n]\!]_\delta)(Z, [\![2k+1]\!]_\delta) \mid \delta \in \Delta\}$. Clearly, $[\![n]\!]_\delta$ is odd for some δ, which means we cannot apply PCNT. This means that the only way to proceed is applying PCC again. But then we have the terms $\text{even}(n)$ and $\text{even}(3k+2)$ that cannot be evaluated. This is what we tried to avoid by choosing our generalisation. The same problem still applies when we use a generalisation such as $Z(2(2n+1)) \subset Z(2k+1)$, only we get 'stuck' after one more application of PCC. So, trying to avoid having to work with $\text{even}(n)$ doesn't work.

What does work is to generalise to $\emptyset \vdash_{pcc} Z(n) \subset Z(k)$ and applying PCC once to get

$$\begin{aligned}\Gamma \vdash_{pcc} &[\text{even}(n) \Rightarrow Z(n \div 2)] \wedge [\neg\,\text{even}(n) \Rightarrow Z((3n+1) \div 2)] \\ &\subset [\text{even}(k) \Rightarrow Z(k \div 2)] \wedge [\neg\,\text{even}(k) \Rightarrow Z((3k+1) \div 2)],\end{aligned}$$

with $\Gamma = \{(Z, [\![n]\!]_\delta) \subset (Z, [\![k]\!]_\delta) \mid \delta \in \Delta\}$. The full derivation is a bit lengthy, so I will only provide a sketch for the rest. What we want to derive is in essence $(\neg x \vee A) \wedge (x \vee B) \subset (\neg y \vee C) \wedge (y \vee D)$, where we know $A \subset C$, $A \subset D$, $B \subset C$ and $B \subset D$ (by applying PCNT). We can derive $(\neg x \vee A) \wedge (x \vee B) \subset A \vee C \vee (x \wedge \neg x)$ using only the inequality logic and negation free propositional logic rules. To derive $(x \wedge \neg x) \subset \bot$, we need the rule DLOGIC. With that, we get $(\neg x \vee A) \wedge (x \vee B) \subset A \vee C$. Next, we can derive $A \vee C \subset (\neg y \vee C) \wedge (y \vee D)$ using inequality logic and negation free propositional logic rules and PCNT. So, with transitivity, we complete the derivation.

The equation system $\mathcal{E}_5$ turns out to be very similar to $\mathcal{E}_1$, in the sense that for all $n \in \mathbb{N}$, the value of $Z(n)$ always depends *only* on $Z(k)$ for some $k \in \mathbb{N}$. The difference is that $\mathcal{E}_1$, this dependence is trivial, while in $\mathcal{E}_5$, it is a consequence of the fact that $\text{even}(n) \wedge \neg\,\text{even}(n) \subset \bot$.

The following equation system is adapted from the previous one:

$$\mathcal{E}_6 = (\mu Z(n : \text{Nat}) = [\text{even}(n) \Rightarrow Z(n \div 2)] \wedge [(\neg\,\text{even}(n) \wedge n > 1) \Rightarrow Z((3n+1) \div 2)]).$$

The variable Z in this system is similar to the previous, although we now have that $Z(1) = \top$ and therefore we cannot relate $Z(1)$ with $Z(n)$ for $n > 1$ via consistent consequence. Here, we cannot use the generalisation to $(Z, n) \subset (Z, k)$, but the generalisation to $(Z, n+2) \subset (Z, k+2)$ would work, as it excludes the case $(Z, 1)$ and we can use DLOGIC to show that this generalisation indeed depends on cases where $(Z, k+2)$.

5.3 Conclusions

As mentioned earlier, whether or not this system is complete mostly depends on whether there exists an equation system such that a consistent consequence relation wouldn't be able to be 'described' with a a finite amount of applications of the rule PCC. In this section, we've seen that we at least need a system where there not every instances of the variables are related via consistent consequence. Additionally, while boolean data terms might prevent some generalisations, they don't prevent all of them. In general, I expect that for equation systems consisting of only one block (so, with only one fixpoint), any complication that a boolean data term introduces can be reduced to statements that exclude any of the relations specific to the equation system and hence can be shown via an application of DLOGIC. I'm uncertain what happens when we introduce multiple blocks. On the one hand, it seems that this allows to make a dependency graph that is more complex and can possible encode some not

Hence, to find an equation system for which completeness can 'fail', I think the likely candidate to consider a system with multiple blocks, where the number of applications of PCC such that a variable gets replaced by one in a block would depend in some non-trivial manner on a boolean data term. However, when working with multiple blocks, I think we'd have to take into account that it also becomes 'less likely' that two variables are related via consistent consequence, as all pairs in a consistent consequence relation that relates them must be from the same block. In other words, while this construction might lead to consistent consequence being 'hard' to derive in this system, it seems to make it 'hard' for the variables to be related in the first place as well. So, I think that while this approach may work, finding such a system can still be quite the challenge.

Finally, note that even if this system turns out to be complete in the restricted setting of this report, this system is not complete for the general case of a PBES. This is because we can make an universal or existential quantification over variables in a PBES and the system from this report doesn't have a rule to even syntactically 'interact' with such a term.

References

[1] MW Gazda. *Fixpoint logic, games, and relations of consequence*. PhD thesis, Technische Universiteit Eindhoven, 2016.

[2] Simona Orzan, Wieger Wesselink, and Tim AC Willemse. Static analysis techniques for parameterised boolean equation systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 230–245. Springer, 2009.

[3] Myrthe van Delft, Herman Geuvers, and Tim AC Willemse. A formalisation of consistent consequence for boolean equation systems. In *International Conference on Interactive Theorem Proving*, pages 462–478. Springer, 2017.