

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

***A Boolean algebra of contracts for logical
assume-guarantee reasoning***

Yann Glouche, Paul Le Guernic, Jean-Pierre Talpin et Thierry Gautier

N° 6570

Juillet 2008

Thèmes COM et SYM



***Rapport
de recherche***

A Boolean algebra of contracts for logical assume-guarantee reasoning

Yann Glouche, Paul Le Guernic, Jean-Pierre Talpin et Thierry Gautier *

Thèmes COM et SYM — Systèmes communicants et Systèmes symboliques
Projets ESPRESSO

Rapport de recherche n° 6570 — Juillet 2008 — 38 pages

Abstract: Assume-guarantee reasoning is a popular and expressive paradigm for a modular and compositional specification of programs. It is in turn of becoming a fundamental concept in mainstream industrial computer-aided design tools for embedded system design. In this paper, we elaborate new foundations for contract-based embedded system design by proposing a general-purpose algebra of assume/guarantee contracts based on two simple concepts: first, the assumption or guarantee of a component is defined as a filter and, second, filters enjoy the structure of a Boolean algebra. This yields an algebraically rich structure which allows us to reason on contracts.

Key-words: assume/guarantee, contract, embedded system, verification, Boolean algebra

* {yann.glouche, paul.leguernic, jean-pierre.talpin, thierry.gautier}@irisa.fr

Une algèbre booléenne de contrats pour un raisonnement logique sur hypothèses/garanties

Résumé : Le raisonnement basé sur hypothèses/garanties est un paradigme populaire et expressif pour la spécification modulaire et compositionnelle de programmes. Cette approche devient un concept fondamental dans l'informatique industrielle des outils de conception assistée par ordinateur pour les systèmes embarqués. Dans ce rapport, nous élaborons de nouvelles bases pour la conception des systèmes embarqués fondée sur les contrats, en proposant une algèbre de contrats générale, basée sur deux concepts simples : d'une part, les hypothèses et garanties d'un composant sont définies en tant que filtres, et d'autre part, les filtres ont une structure d'algèbre booléenne. Il en résulte une structure algébrique riche qui permet de raisonner sur les contrats.

Mots-clés : hypothèse/garantie, contrat, système embarqué, vérification, algèbre booléenne

Contents

1	Introduction	4
2	An algebra of processes	4
3	An algebra of filters	9
4	An algebra of contracts	14
5	Related work	18
6	Discussion	18
7	Conclusion	20
	References	22
A	Proofs of Section 2	23
B	Proofs of Section 3	25
C	Proofs of Section 4	30

1 Introduction

Common methodological precepts for attacking the design of large embedded architectures advise the validation of specifications as early as possible and an iterative validation of each refinement or modification made to the initial specification, until the implementation of the system is finalized. Additionally, cooperative component-based development requires to use and to assemble components, that have been developed by different suppliers, and in a safe and consistent way. These components have to be provided with their conditions of use and some guarantees that they have been validated when these conditions are satisfied.

We adopt the paradigm of *contract* to define a component-based validation process in the context of a synchronous modeling framework. We define a novel algebraic framework to enable logical reasoning on contracts. It is based on two simple concepts. First, the assumptions and guarantees of a component are defined as filters: assumptions filter the behavior a component may accept and guarantees filter the behaviors a component provides. Second and foremost, we define a Boolean algebra to manipulate filters. This yields an algebraically rich structure which allows us to reason on contracts (to abstract, refine, combine and normalize them). This algebraic model is based on a minimalist model of execution traces, allowing one to adapt it easily to a particular design framework.

The most important aspect introduced by the framework is the notion of filter, for which the negation is clearly defined. A filter constrains a finite set of variables, which is represented by the set of the processes which satisfy these constraints.

Plan The paper is organized as follows. Section 2 introduces a suitably general algebra of processes which borrows its notation and concepts to domain theory [10]. A contract (\mathbf{A}, \mathbf{G}) is viewed as a pair of logical devices filtering processes: the assumption \mathbf{A} filters processes to select (accept or conversely reject) those that are asserted (accepted or conversely rejected) by the guarantee \mathbf{G} . Process-filters are defined in Section 3 and contracts in Section 4. Section 5 presents related work and which is further discussed around an example in Section 6. Section 7 concludes the presentation.

2 An algebra of processes

We start with the definition of a suitable algebra for behaviors and processes. Usually, a behavior describes the trace of a discrete process (a Mazurkiewicz trace or a tuple of signals in Lee's tagged signal model). We deliberately choose a more abstract definition in order to encompass not only discrete behaviors on Boolean, integer, real variables but also behaviors of more complex systems, such as continuous functions.

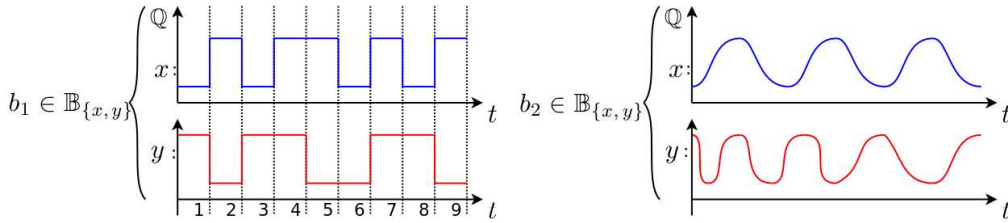
Definition 1. [Behavior] Let \mathcal{V} be an infinite, countable set of variables, and \mathcal{D} a set of values; for \mathbf{Y} , a finite set of variables included in \mathcal{V} (written $\mathbf{Y} \subset \mathcal{V}$), \mathbf{Y} nonempty, a *\mathbf{Y} -behavior* is a function $c : \mathbf{Y} \rightarrow \mathcal{D}$; the set of \mathbf{X} -behaviors is $\mathbb{B}_{\mathbf{X}}$. This is extended to the

empty variable domain:

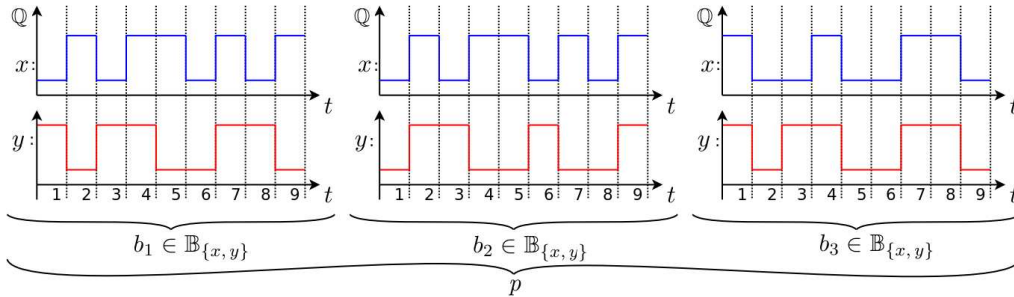
$$\mathbb{B}_{\mathbf{Y}} =_{\Delta} \mathbf{Y} \rightarrow \mathcal{D} \text{ and } \mathbb{B}_{\emptyset} =_{\Delta} \emptyset$$

For \mathbf{Y} , a finite set of variables included in \mathcal{V} , \mathbf{Y} nonempty, c a \mathbf{Y} -behavior, \mathbf{X} a (possibly empty) subset of \mathbf{Y} , $c|_{\mathbf{X}}$ is the \mathbf{X} -behavior equal to c on \mathbf{X} :

$$c|_{\mathbf{X}} =_{\Delta} \{(x, c(x)) / x \in \mathbf{X}\} \text{ and } c|_{\emptyset} = \emptyset \text{ and } c|_{\mathbf{Y}} = c \tag{1}$$



From left to right – The x, y -behaviors b_1 and b_2 are functions from the variables x, y to functions that denote signals. Left, behavior b_1 is a discrete sampling mapping a domain of time represented by natural numbers to values in rationals \mathbb{Q} . Right, behavior b_2 associates x, y to continuous functions of time. A *process* is denoted by a set of behaviors on a given set of variables. For instance, the process of behavior b_1 , below, contains other possible behaviors on the variables x and y .



Definition 2. [Process] For \mathbf{X} , a finite set of variables ($\mathbf{X} \subset \mathcal{V}$), an \mathbf{X} -process p is a nonempty set of \mathbf{X} -behaviors.

Thus, since $\mathbb{B}_{\emptyset} =_{\Delta} \emptyset$, there is a unique \emptyset -process designated by $\Omega =_{\Delta} \{\emptyset\}$; Ω has the empty behavior as unique behavior. The *empty process* is denoted by $\mathcal{U} =_{\Delta} \emptyset$.

Since Ω does not have any variable, it has no effect when composed (intersected) with other processes. It can be seen as the universal process, for constraint conjunction, in

contrast with \mathcal{U} , the empty set of behaviors, the use of which in constraint conjunction always results in the empty set. \mathcal{U} can be seen as the null process.

For \mathbf{X} , a finite set of variables ($\mathbf{X} \subset \mathcal{V}$), we denote by $\mathbb{P}_{\mathbf{X}}$ the set of \mathbf{X} -processes. A process in $\mathbb{P}_{\mathbf{X}}$ defined on a finite set of variables \mathbf{X} is said *strict* (thus Ω is a strict process). \mathbb{P} denotes the set of all strict processes.

$$\mathbb{P}_{\mathbf{X}} =_{\Delta} \mathcal{P}(\mathbb{B}_{\mathbf{X}}) \setminus \{\mathcal{U}\}, \quad \mathbb{P} =_{\Delta} \cup_{(\mathbf{X} \subset \mathcal{V})} \mathbb{P}_{\mathbf{X}} \quad (\mathbb{P}_{\emptyset} = \{\Omega\})$$

The domain of behaviors in an \mathbf{X} -process p is denoted by $var(p) =_{\Delta} \mathbf{X}$. \mathcal{U} is the only non-strict \mathcal{V} -process : $var(\mathcal{U}) = \mathcal{V}$. A process is either \mathcal{U} , or a strict process. Hence, the set of all processes \mathbb{P}^* is defined by $\mathbb{P}^* =_{\Delta} \mathbb{P} \cup \{\mathcal{U}\}$ and $\forall \mathbf{X} \subset \mathcal{V}, \mathbb{P}^*_{\mathbf{X}} =_{\Delta} \mathbb{P}_{\mathbf{X}} \cup \{\mathcal{U}\}$. For $\mathbf{R} \subseteq \mathbb{P}^*$, $\overline{\mathbf{R}}$ denotes the complementary of \mathbf{R} . We define the complementary of a process and its restriction or extension of the behaviors to a given set of variables.

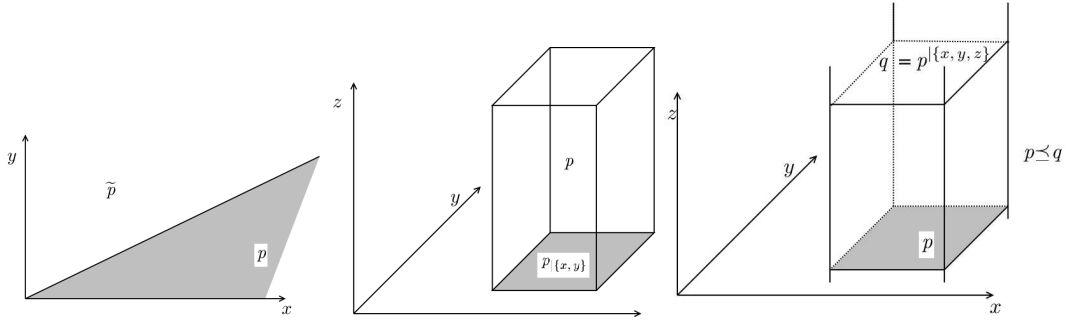
Definition 3. [Complementary of a process] For \mathbf{X} , a finite set of variables ($\mathbf{X} \subset \mathcal{V}$), the complementary \tilde{p} of a process $p \in \mathbb{P}_{\mathbf{X}}$ is defined by:

$$p \in \mathbb{P}_{\mathbf{X}} \implies \tilde{p} =_{\Delta} (\mathbb{B}_{\mathbf{X}} \setminus p) = \{b \in \mathbb{B}_{\mathbf{X}} / b \notin p\} \quad (\widetilde{\mathbb{B}_{\mathbf{X}}} = \mathcal{U}) \quad (2)$$

Definition 4. [Process restriction and extension] When \mathbf{X}, \mathbf{Y} are finite sets of variables such that $\mathbf{X} \subseteq \mathbf{Y} \subset \mathcal{V}$, \mathbf{Y} nonempty, we define the restriction $q|_{\mathbf{X}} \in \mathbb{P}_{\mathbf{X}}$ of $q \in \mathbb{P}_{\mathbf{Y}}$ to \mathbf{X} and conversely the extension $p|_{\mathbf{Y}} \in \mathbb{P}_{\mathbf{Y}}$ of $p \in \mathbb{P}_{\mathbf{X}}$ to \mathbf{Y} by:

$$q|_{\mathbf{X}} =_{\Delta} \{c|_{\mathbf{X}} / c \in q\} \quad (\text{then } q|_{\emptyset} = \Omega, q|_{var(q)} = q) \quad (3)$$

$$p|_{\mathbf{Y}} =_{\Delta} \{c \in \mathbb{B}_{\mathbf{Y}} / c|_{\mathbf{X}} \in p\} \quad (\text{then } \Omega|_{\mathbf{Y}} = \mathbb{B}_{\mathbf{Y}}, p|_{var(p)} = p) \quad (4)$$



Left, the complementary \tilde{p} of a process p defined on the variables x and y consists of all behaviors defined on x, y not belonging to p – Center, the restriction $p|_{\{x, y\}}$ of a

process p defined on x, y, z consists of its projection on the restricted domain – Right, the extension $p^{|x, y, z\}$ of a process p defined on x, y is the largest process defined on x, y, z whose restriction on x, y is equal to p .

The set $\mathbb{P}^*_{\mathbf{X}}$, equipped with union, intersection and complementary is a Boolean algebra with supremum $\mathbb{P}^*_{\mathbf{X}}$ and infimum \mathcal{U} . Restriction is extended to \mathcal{U} , the blocking process, by $\mathcal{U}|_{\mathbf{X}} = \{c|_{\mathbf{X}} \mid c \in \emptyset\} = \mathcal{U}$. The restriction and extension of strict processes satisfy the following properties.

Property 1. When $\mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are finite sets of variables, \mathbf{Y}, \mathbf{Z} nonempty, p, q strict processes:

$$\text{var}(p) \subseteq \mathbf{Z} \subseteq \mathbf{Y} \implies (p^{|z|_{\mathbf{Y}}} = p^{|y|_{\mathbf{Y}}}) \wedge (p^{|y|_{\mathbf{Z}}} = p^{|z|_{\mathbf{Z}}}) \quad (5)$$

$$\text{var}(p) = \text{var}(q) \subseteq \mathbf{Y} \implies (((p \cap q)^{|y|_{\mathbf{Y}}} = (p^{|y|_{\mathbf{Y}}} \cap q^{|y|_{\mathbf{Y}}})) \wedge ((p \cup q)^{|y|_{\mathbf{Y}}} = (p^{|y|_{\mathbf{Y}}} \cup q^{|y|_{\mathbf{Y}}})) \quad (6)$$

$$\text{var}(p) = \text{var}(q) \subseteq \mathbf{Y} \implies ((p \subseteq q) \iff (p^{|y|_{\mathbf{Y}}} \subseteq q^{|y|_{\mathbf{Y}}})) \quad (7)$$

$$\mathbf{X} \subseteq \text{var}(p) = \text{var}(q) \implies ((p \subseteq q) \implies (p|_{\mathbf{X}} \subseteq q|_{\mathbf{X}})) \quad (8)$$

We define the poset of strict processes.

Definition 5. [Strict processes extension] For nonempty finite sets of variables $\mathbf{X} \subseteq \mathbf{Y} \subseteq \mathcal{V}$ and for $p \in \mathbb{P}^*_{\mathbf{X}}$, the relation $p \preceq q$ means that q is an extension of p to \mathbf{Y} :

$$(p \preceq q) \iff ((\text{var}(p) \subseteq \text{var}(q)) \wedge (p^{\text{var}(q)} = q))$$

Property 2. (\mathbb{P}, \preceq) is a poset.

The upper set $[\preceq \uparrow p]$ of a process p is the set of all its extensions:

$$[\preceq \uparrow p] =_{\Delta} \{q \in \mathbb{P} / p \preceq q\} \quad ([\preceq \uparrow \Omega] = \{\mathbb{B}_{\mathbf{X}}\}_{\mathbf{X} \subseteq \mathcal{V}}) \quad (9)$$

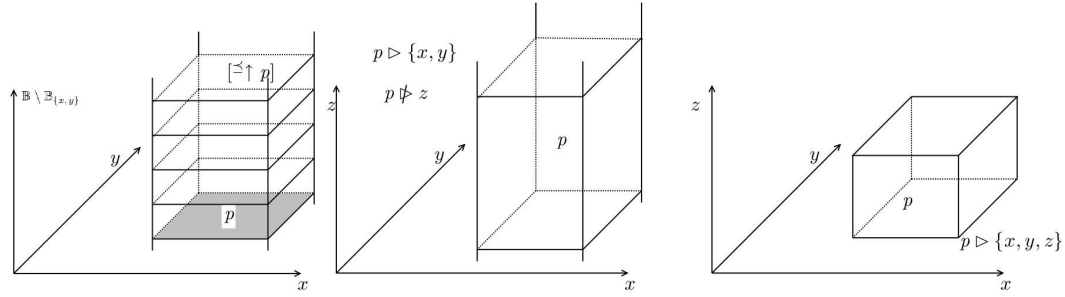
Definition 6. [Variable control \triangleright] A process q controls a variable y , written $(q \triangleright y)$, iff

$$((y \in \text{var}(q)) \wedge q \not\subseteq ((q|_{(\text{var}(q) \setminus \{y\})})^{\text{var}(q)})) \quad (10)$$

A process q controls a variable set X , written $(q \triangleright X)$ iff

$$(\forall x \in \mathbf{X})(q \triangleright x) \quad (\Omega \triangleright \emptyset) \quad (11)$$

Moreover, \triangleright is extended to \mathbb{P}^* with $\mathcal{U} \triangleright \mathcal{V}$.



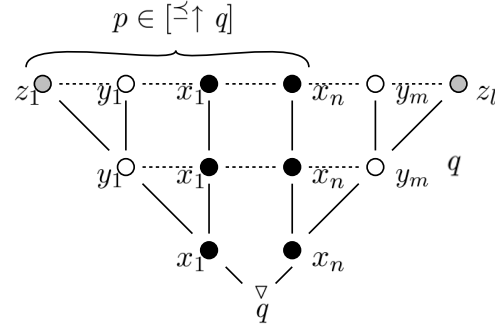
Left, the upper set $[\geq\uparrow p]$ is the set of all processes $q \in \mathbb{P}$ such that $(p \leq q)$ – Center, let $\mathbf{X} = \{x, y, z\}$, a process $p \in \mathbb{P}_{\mathbf{X}}$ controls the variables x and y and lets z free – Right, a process $p \in \mathbb{P}_{\mathbf{X}}$ controls the variables x, y, z .

Note that, if a process p controls \mathbf{X} , this does not imply that, for all $x \in \mathbf{X}$, $y \in \mathbf{X}$, $x \neq y$, $(p|_{(\mathbf{X} \setminus \{x\})})$ controls y .

Definition 7. [Reduced process] A strict process p is *reduced* iff it controls all its variables: p is *reduced* iff $p \triangleright \text{var}(p)$.

For instance, Ω is reduced. *Reduced strict processes* are minimal in (\mathbb{P}, \leq) . We denote by $\overset{\vee}{q}$, called *reduction of q* , the (minimal) strict process such that $\overset{\vee}{q} \leq q$ (p is reduced iff $\overset{\vee}{p} = p$).

Right, the reduction $\overset{\vee}{q}$ of a process q and a process p in the upper set $[\geq\uparrow q]$. Assuming that $\text{var}(q) = (\{x_1 \dots x_n\} \cup \{y_1 \dots y_m\})$ and that q controls the variables $\{x_1 \dots x_n\}$, we have $\text{var}(\overset{\vee}{q}) = \{x_1 \dots x_n\}$. The process p is such that $p \in [\geq\uparrow \overset{\vee}{q}]$ with $\text{var}(p) \subseteq (\{x_1 \dots x_n\} \cup \{y_1 \dots y_m\} \cup \{z_1 \dots z_l\})$. Process p controls the variables $\{x_1 \dots x_n\}$, and $\{y_1 \dots y_m\} \cup \{z_1 \dots z_l\}$ is a set of free variables, such that $\overset{\vee}{q} = \overset{\vee}{p}$.



Property 3. The complementary \tilde{p} of a strict process p is reduced iff p is reduced; \tilde{p} and p control the same set of variables $\text{var}(p)$.

From the above, we deduce that $[\geq\uparrow \overset{\vee}{p}]$, the upper set of the *reduction of p* , is a (principal) filtered set [10]: it is nonempty and each pair of elements has a lower bound. We also observe that $\text{var}(\overset{\vee}{q})$ is the greatest subset of variables such that $q \triangleright \text{var}(\overset{\vee}{q})$; for a strict process q , we extend the definition of $\text{var}()$ to the upper set of its *reduction* by $\text{var}([\geq\uparrow \overset{\vee}{q}]) =_{\Delta} \text{var}(\overset{\vee}{q})$.

Property 4. The upper set of a strict process p contains a unique process $p^{\mathbf{Y}}$ defined on a given set of variables $\mathbf{Y} \supseteq \text{var}(p)$; the process p and its extension $p^{\mathbf{Y}}$ control the same set of variables, that is the set of variables controlled by the reduction of p .

$$((q \in [\preceq \uparrow p]) \wedge (r \in [\preceq \uparrow p]) \wedge (\text{var}(q) = \text{var}(r))) \implies (q = r) \quad (12)$$

$$(\text{var}(p) \subseteq \mathbf{Y}) \implies ((p^{\mathbf{Y}}) \triangleright \text{var}(\bar{p})) \quad (13)$$

$$((\text{var}(p) \cup \text{var}(q)) \subseteq \mathbf{Y}) \implies ((p^{\mathbf{Y}} = q^{\mathbf{Y}}) \implies (\bar{p} = \bar{q})) \quad (14)$$

For the blocking process, we set $\mathcal{U} \triangleright \mathcal{V}$ and $[\preceq \uparrow \mathcal{U}] = \{\mathcal{U}\}$.

We define the *inclusion lower set* of a process to capture all the subsets of its behaviors. Let $\mathbf{R} \subseteq \mathbb{P}^*$, $[\mathbf{R}]_{\sqsubseteq}$ is the lower set of \mathbf{R} for \sqsubseteq :

$$[\mathbf{R}]_{\sqsubseteq} =_{\Delta} \{p \in \mathbb{P}^* / (\exists q \in \mathbf{R})(p \subseteq q)\} \quad (15)$$

Property 5. From the above definitions, we conclude that:

$$[[\preceq \uparrow \bar{\mathcal{U}}]_{\sqsubseteq}] = \{\mathcal{U}\} \quad (16)$$

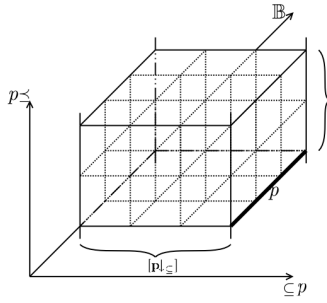
$$[[\preceq \uparrow \bar{\Omega}]_{\sqsubseteq}] = \mathbb{P}^* \quad (17)$$

3 An algebra of filters

In this section, we define a *process-filter* by the set of processes that satisfy a given property. We propose an order relation (\sqsubseteq) on the set of process-filters Φ . We establish that (Φ, \sqsubseteq) is a lattice and a Boolean algebra. A *process-filter* \mathbf{R} is a subset of \mathbb{P}^* that filters processes. It contains all processes that are “equivalent” with respect to some constraint or property, so that all processes in \mathbf{R} are accepted or all of them but \mathcal{U} are rejected. A process-filter is built from a unique process *generator* by extending it to larger sets of variables, and then by including subprocesses of these “maximal allowed behavior sets”.

Definition 8. [Process-filter] A set of processes \mathbf{R} is a *process-filter* iff $(\exists r \in \mathbb{P}^*) (((r = \bar{r}) \wedge (\mathbf{R} = [[\preceq \uparrow r]_{\sqsubseteq}])))$. The process r is a *generator* of \mathbf{R} (\mathbf{R} is generated by r). We denote by Φ is the set of process-filters.

The process-filter generated by the reduction of a process p is denoted by $\widehat{p} =_{\Delta} [[\uparrow^{\nabla} p]_{\downarrow \subseteq}]$.



Left, a process-filter is generated from the process p (depicted by a bold line) via two successive operations. The first operation consists of building the upper set of the process: takes all the processes that are compatible with p and that are defined on a bigger set of variables. The second operation proceeds using the inclusion lower set of this set of processes: it takes all the processes that are defined by subsets of behaviors from processes in the upper set (in other words, those processes that remain compatible when adding constraints, because adding constraints removes behaviors).

A process-filter $\mathbf{R} = \widehat{r}$ satisfies the following properties:

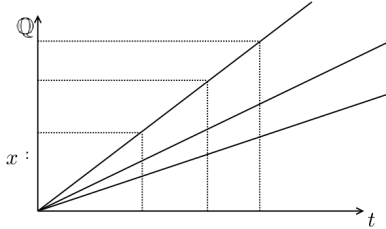
Property 6. The variable set of a process p , that belongs to a process-filter generated by a reduced process \overline{r} , contains the variable set of this process \overline{r} . The generator of a process-filter is unique; we refer to it as \mathbf{R} . Finally Ω generates the set of all processes (including \mathcal{U}), \mathcal{U} belongs to all filters. Formally ($\forall p, r, s \in \mathbb{P}^*$):

$$(p \in \widehat{r}) \implies (\text{var}(\overline{r}) \subseteq \text{var}(p)) \quad (18)$$

$$\widehat{r} = \widehat{s} \iff \overline{r} = \overline{s} \quad (19)$$

$$\Omega \in \widehat{r} \iff \widehat{r} = \mathbb{P}^* \quad (20)$$

$$\mathcal{U} \in \mathbf{R} \quad (21)$$



Let $p \in \mathbb{P}_{\{x\}}$ be a process defined on $x \in \mathcal{V}$ a variable whose behaviors are a function from a totally ordered domain of time \mathbb{T} to rationals \mathbb{Q} . Define the process-filter p to satisfy :

$$\forall b \in p, \quad b(x) : \mathbb{T} \mapsto \mathbb{Q} \\ \forall t, t' \in \mathbb{T}, t \leq t' \iff b(x)(t) \leq b(x)(t')$$

Then \widehat{p} is the set of all processes s.t. $\forall b \in \mathbb{B}, b \in p$, $b(x)$ is monotonic increasing function from the domain of time \mathbb{T} to \mathbb{Q} .

We call *strict process-filters* the process-filters that are neither \mathbb{P}^* nor $\{\mathcal{U}\}$. The filtered variable set of \mathbf{R} is $var(\mathbf{R})$ defined by:

$$var(\mathbf{R}) =_{\Delta} var(\overset{\nabla}{\mathbf{R}}) \quad (22)$$

Theorem 1. A strict process p belongs to a process-filter \mathbf{R} iff

$$(\forall \mathbf{X}, \mathbf{Y} \subseteq \mathcal{V})(var(\mathbf{R}) \subseteq \mathbf{X} \subseteq \mathbf{Y}), (p \in \mathbf{R}) \iff \begin{cases} (var(\mathbf{R}) \subseteq var(p)) \\ (var(p) \subseteq \mathbf{Y}) \implies ((p|_{\mathbf{Y}})|_{\mathbf{X}} \subseteq \overset{\nabla}{\mathbf{R}}|_{\mathbf{X}}) \end{cases}$$

Corollary 1. The two equivalent properties are satisfied:

$$\mathbf{R} \subseteq \mathbf{S} \iff ((var(\mathbf{S}) \subseteq var(\mathbf{R})) \wedge (\overset{\nabla}{\mathbf{R}}|_{var(\mathbf{S})} \subseteq \overset{\nabla}{\mathbf{S}})) \quad (23)$$

$$\mathbf{R} \subseteq \mathbf{S} \iff \overset{\nabla}{\mathbf{R}} \in \mathbf{S} \quad (24)$$

Corollary 2. The following properties are satisfied:

$$(\mathbf{R} \subseteq (\mathbf{S} \cap \mathbf{T})) \iff ((\mathbf{R} \subseteq \mathbf{S}) \wedge (\mathbf{R} \subseteq \mathbf{T})) \text{(corollary 1 – equation 24)} \quad (25)$$

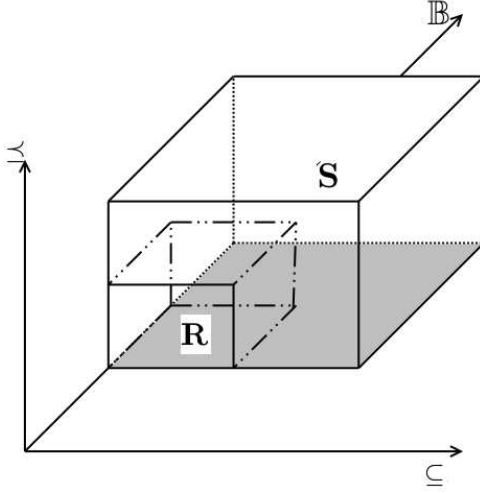
$$(\mathbf{R} \subseteq (\mathbf{S} \cup \mathbf{T})) \iff ((\mathbf{R} \subseteq \mathbf{S}) \vee (\mathbf{R} \subseteq \mathbf{T})) \text{(corollary 1 – equation 24)} \quad (26)$$

We define an order relation on process-filters, which we call relaxation, and write $\mathbf{R} \sqsubseteq \mathbf{S}$ to mean that \mathbf{R} is less wide than \mathbf{S} .

Definition 9. [Process-filter relaxation] For \mathbf{R} and \mathbf{S} , two process-filters, the relation \mathbf{R} is less wide than \mathbf{S} , written $\mathbf{R} \sqsubseteq \mathbf{S}$ is defined by:

$$\{\mathcal{U}\} \sqsubseteq \mathbf{S} \quad (\mathbf{R} \sqsubseteq \{\mathcal{U}\}) \iff \{\mathcal{U}\} = \mathbf{R} \quad (\mathbf{R} \sqsubseteq \mathbf{S} \iff \overset{\nabla}{\mathbf{R}}|_{\mathbf{Z}} \subseteq \overset{\nabla}{\mathbf{S}}|_{\mathbf{Z}}) \quad (27)$$

where $\mathbf{Z} = var(\mathbf{R}) \cup var(\mathbf{S})$



Left, the relation between two process-filters \mathbf{R} and \mathbf{S} which represent the same set of constraints but on a different set of controlled variables: \mathbf{R} is less wide than \mathbf{S} because the set of processes represented by \mathbf{R} under the given constraints is included in that of \mathbf{S} .

For instance, let $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ the filters respectively generated by the constraints $(x \in \{0, 1\} \wedge y = 1)$, $x \in \{0, 1\}$, $(x \in \{0, 1\} \vee (x = 2 \wedge z = 0))$; they satisfy $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2 \sqsubseteq \mathbf{R}_3$. We have that $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ and meanwhile:

- $\mathbf{R}_1 \not\sqsubseteq \mathbf{R}_3$ and $\mathbf{R}_1 \not\sqsubseteq \mathbf{R}_3$
- $\mathbf{R}_2 \not\sqsubseteq \mathbf{R}_3$ and $\mathbf{R}_2 \not\sqsubseteq \mathbf{R}_3$

Property 7. Strict process-filters \mathbf{R} and \mathbf{S} satisfy $(\mathbf{R} \sqsubseteq \mathbf{S}) \iff ((\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge \mathbf{R} \sqsubseteq \mathbf{S})$

The relaxation relation defines the structure of process-filters, which is shown to be a lattice.

Property 8. (Φ, \sqsubseteq) is a poset.

Lemma 1. (Φ, \sqsubseteq) is a lattice with \mathbb{P}^* as supremum and $\{\emptyset\}$ as infimum; the infimum (or conjunction) $\mathbf{R} \sqcap \mathbf{S}$, the supremum (or disjunction) $\mathbf{R} \sqcup \mathbf{S}$ are defined by:

$$\{\emptyset\} \sqcap \mathbf{R} = \mathbf{R} \sqcap \{\emptyset\} = \{\emptyset\} \quad (28)$$

$$(\mathbf{R} \neq \{\emptyset\} \wedge \mathbf{S} \neq \{\emptyset\}) \implies \mathbf{R} \sqcap \mathbf{S} =_{\Delta} [[\overset{\nabla}{\neq} \uparrow \overset{\nabla}{p}] \downarrow_{\subseteq}] \quad (29)$$

where $p = (\overset{\nabla}{\mathbf{R}} \cap \overset{\nabla}{\mathbf{S}})$, $\mathbf{V} = \text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S})$

$$\{\emptyset\} \sqcup \mathbf{R} = \mathbf{R} \sqcup \{\emptyset\} = \mathbf{R} \quad (30)$$

$$(\mathbf{R} \neq \{\emptyset\} \wedge \mathbf{S} \neq \{\emptyset\}) \implies \mathbf{R} \sqcup \mathbf{S} =_{\Delta} [[\overset{\nabla}{\neq} \uparrow \overset{\nabla}{p}] \downarrow_{\subseteq}] \quad (31)$$

where $p = (\overset{\nabla}{\mathbf{R}} \cup \overset{\nabla}{\mathbf{S}})$, $\mathbf{V} = \text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S})$

Lemma 2. The following properties hold: for $\mathbf{R}, \mathbf{S}, \mathbf{T}$ strict process-filters,

$$((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \sqsubseteq (\text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T})) \quad (32)$$

$$\text{and thus } ((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq (\mathbf{R} \sqcap \mathbf{S}))(\text{take } (\mathbf{R} \sqcap \mathbf{S}) = \mathbf{T}) \quad (33)$$

$$((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \sqsubseteq (\text{var}(\mathbf{R}) \cap \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T})) \quad (34)$$

$$\text{and thus } ((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq (\mathbf{R} \sqcup \mathbf{S}))(\text{take } (\mathbf{R} \sqcup \mathbf{S}) = \mathbf{T}) \quad (35)$$

Definition 10. [Process-filter complementary] The complementary $\widetilde{\mathbf{R}}$ of a process-filter \mathbf{R} is defined by:

$$\{\widetilde{\mathcal{U}}\} = \mathbb{P}^*, \widetilde{\mathbb{P}^*} = \{\mathcal{U}\} \quad (36)$$

$$(\mathbf{R} \neq \{\mathcal{U}\} \wedge \mathbf{R} \neq \mathbb{P}^*) \implies (\widetilde{\mathbf{R}} =_{\Delta} [[\overset{\nabla}{\uparrow} \mathbf{R}] \downarrow_{\sqsubseteq}]) \quad (37)$$

If $\mathbf{R} \neq \{\mathcal{U}\}$ and $\widetilde{\mathbf{R}} \neq \{\mathcal{U}\}$ then $\widetilde{\mathbf{R}} = (\mathbb{B}_{\text{var}(\mathbf{R})} \setminus \overset{\nabla}{\mathbf{R}})$ is reduced and $\text{var}(\mathbf{R}) = \text{var}(\widetilde{\mathbf{R}})$ (see equation 2 and property 3).

Corollary 3. The complementary of a filter \mathbf{R} satisfies $\widetilde{\mathbf{R}} \sqsubseteq \overline{\mathbf{R}} \cup \{\mathcal{U}\}$.

We formalize our main result, which is that process-filters form a Boolean algebra.

Theorem 2. [Process-filter Boolean algebra] (Φ, \sqsubseteq) is a Boolean algebra with \mathbb{P}^* as 1, $\{\mathcal{U}\}$ as 0 and the complementary $\widetilde{\mathbf{R}}$.

The process-filter conjunction $\mathbf{R} \sqcap \mathbf{S}$ of two strict process-filters \mathbf{R} and \mathbf{S} is the greatest process-filter $\mathbf{T} = \mathbf{R} \sqcap \mathbf{S}$ that accepts all processes that are accepted by \mathbf{R} and by \mathbf{S} .

Example. Let x , a variable taking values in $\{0,1,2,3\}$ and u, y, v three variables taking values in $\{0,1\}$; let $r \in \mathbb{P}_{\{u, x, y\}}$, $s \in \mathbb{P}_{\{x, y, v\}}$, two reduced processes defined by

$$\begin{aligned} r &= \{b/b(u) \in \{0,1\} \wedge b(x) \in \{0,1\} \wedge b(y) \in \{0,1\}\} \cup \{(u,1), (x,2), (y,0)\} \\ s &= \{b/b(x) \in \{0,1\} \wedge b(y) \in \{0,1\} \wedge b(v) \in \{0,1\}\} \cup \{(x,3), (y,1), (v,0)\} \end{aligned}$$

One can see that $r \triangleright \{u, x, y\}$; u and y are free in r when x is 0 or 1; v is free whatever the value of x is in r . We also have $s \triangleright \{x, y, v\}$; y and v are free in s when x is 0 or 1; thus u is free whatever the value of x is in s . From the above definitions, we have that $p =_{\Delta}$

$r \cap s = \{b \mid b(u) \in \{0,1\} \wedge b(x) \in \{0,1\} \wedge b(y) \in \{0,1\} \wedge b(v) \in \{0,1\}\}$ and $\overline{p} = \{b \mid b(x) \in \{0,1\}\}$.

The process-filter disjunction $\mathbf{R} \sqcup \mathbf{S}$ of two strict process-filters \mathbf{R} and \mathbf{S} is the smallest process-filter $\mathbf{T} = \mathbf{R} \sqcup \mathbf{S}$ that accepts all processes that are accepted by \mathbf{R} or by \mathbf{S} .

Example. Let x , a variable taking values in $\{0,1,2,3\}$ and u, y, v three variables taking values in $\{0,1\}$; let $r \in \mathbb{P}_{\{u, x, y\}}$, $s \in \mathbb{P}_{\{x, y, v\}}$, two reduced processes such that

$$\begin{aligned} r &= \{b \mid b(u) \in \{0,1\} \wedge b(x) \in \{0,1\} \wedge b(y) = 0\} \\ s &= \{b \mid b(x) \in \{0,1\} \wedge b(y) = 1 \wedge b(v) \in \{0,1\}\} \end{aligned}$$

hence $p =_{\Delta} r \cup s = \{b \mid b(u) \in \{0,1\} \wedge b(x) \in \{0,1\} \wedge b(y) \in \{0,1\} \wedge b(v) \in \{0,1\}\}$ and $\overline{p} = \{b \mid b(x) \in \{0,1\}\}$.

Definition 11. [Variable elimination in process-filter] Let x a variable, \mathbf{R} a process-filter, and $\mathbf{X} =_{\Delta} \text{var}(\mathbf{R})$, $\mathbf{R}_{|\exists x}$, the E-elimination of x in \mathbf{R} , and $\mathbf{R}_{|\forall x}$, the U-elimination of x in \mathbf{R} , are defined by $\mathbf{R}_{|\exists x} = \widehat{[(\mathbf{R})_{|\mathbf{X} \setminus \{x\}}]}$, when $x \in \mathbf{X}$ and by $\mathbf{R}_{|\exists x} = \mathbf{R}$ otherwise. Also, $\mathbf{R}_{|\forall x} =_{\Delta} \widetilde{\mathbf{R}_{|\exists x}}$

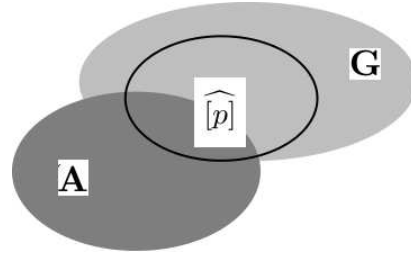
Property 9. $\mathbf{R}_{|\forall x} \sqsubseteq \mathbf{R} \sqsubseteq \mathbf{R}_{|\exists x}$

4 An algebra of contracts

We define the notion of contract and propose an equivalence relation between contracts.

Definition 12. [Contract] A *contract* $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ is a pair of process-filters. $\text{var}(\mathbf{C})$, the variable set of $\mathbf{C} = (\mathbf{A}, \mathbf{G})$, is defined by $\text{var}(\mathbf{C}) = \text{var}(\mathbf{A}) \cup \text{var}(\mathbf{G})$. $\mathbb{C} = \Phi \times \Phi$ is the set of contracts.

Usually, an assumption \mathbf{A} is an assertion on the behavior of the environment (it is typically expressed on the inputs of p), and thus defines the set of behaviors that a process has to take into account. The guarantee \mathbf{G} defines properties that should be guaranteed by a process running in an environment where behaviors satisfy \mathbf{A} . The figure depicts a process p satisfying the contract (\mathbf{A}, \mathbf{G}) .



Definition 13. [Satisfaction] Let $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ a contract, p a process:
 $p \models \mathbf{C} \iff (\widehat{p} \sqcap \mathbf{A}) \sqsubseteq \mathbf{G}$.

Corollary 4. $p \models \mathbf{C} \iff \widehat{p} \sqsubseteq (\widetilde{\mathbf{A}} \sqcup \mathbf{G})$

We define a preorder relation that allows to compare contracts.

Definition 14. [Satisfaction preorder] A contract $(\mathbf{A}_1, \mathbf{G}_1)$ is *finer* than a contract $(\mathbf{A}_2, \mathbf{G}_2)$, written $(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2)$, iff all processes that satisfy the contract $(\mathbf{A}_1, \mathbf{G}_1)$ also satisfy the contract $(\mathbf{A}_2, \mathbf{G}_2)$:

$$(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2) \iff (\forall p \in \mathbb{P})((p \models (\mathbf{A}_1, \mathbf{G}_1)) \implies (p \models (\mathbf{A}_2, \mathbf{G}_2))) \quad (38)$$

Lemma 3. The relation *finer* on contracts satisfies the following property:

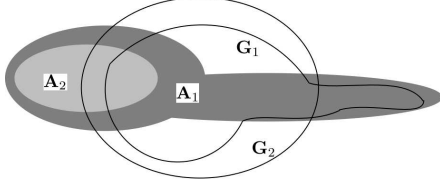
$$(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2) \iff (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqsubseteq (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2) \quad (39)$$

The following relation makes equivalent those contracts that accept the same set of processes.

Definition 15. [Filtering equivalence of contracts] Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ are *filtering equivalent*, denoted $(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow\rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2)$ if and only if:
 $((\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2)) \wedge ((\mathbf{A}_2, \mathbf{G}_2) \rightsquigarrow (\mathbf{A}_1, \mathbf{G}_1))$.

Corollary 5. Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ are *filtering equivalent* if and only if $(\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) = (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)$.

Refinement of contracts amounts to relaxing assumptions and reinforcing promises under the initial assumptions. The intuitive meaning is that for any p that satisfies a contract \mathbf{C} , if \mathbf{C} refines \mathbf{D} then p satisfies \mathbf{D} . Our relation of refinement formalizes substitutability for contracts.



Left, for instance, a contract $(\mathbf{A}_1, \mathbf{G}_1)$ refines a contract $(\mathbf{A}_2, \mathbf{G}_2)$. Among filtering equivalent contracts that can be used to refine an existing contract $(\mathbf{A}_2, \mathbf{G}_2)$, we choose those contracts $(\mathbf{A}_1, \mathbf{G}_1)$ that “scan” more processes than $(\mathbf{A}_2, \mathbf{G}_2)$ ($\mathbf{A}_2 \sqsubseteq \mathbf{A}_1$) and that guarantee less processes than those of $\mathbf{A}_1 \sqcup \mathbf{G}_2$. But other choices could have been made.

Definition 16. [Refinement of contracts] Let $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ two contracts. The contract \mathbf{C}_1 *refines* the contract \mathbf{C}_2 , written $\mathbf{C}_1 \preceq \mathbf{C}_2$, if and only if the three following properties are satisfied (i) $(\mathbf{A}_1, \mathbf{G}_1) \sim (\mathbf{A}_2, \mathbf{G}_2)$ (a) $\mathbf{A}_2 \sqsubseteq \mathbf{A}_1$ and (c) $\mathbf{G}_1 \sqsubseteq \mathbf{A}_1 \sqcup \mathbf{G}_2$.

Lemma 4. $(\mathbf{A}_1, \mathbf{G}_1) \preceq (\mathbf{A}_2, \mathbf{G}_2)$ iff the three following properties are satisfied: (a) $\mathbf{A}_2 \sqsubseteq \mathbf{A}_1$ (b) $(\mathbf{A}_2 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_2$ and (c) $\mathbf{G}_1 \sqsubseteq \mathbf{A}_1 \sqcup \mathbf{G}_2$.

Property 10. (\mathbf{C}, \preceq) is a poset.

The refinement relation (\preceq) defines the poset of contracts, which is shown to be a lattice.

Lemma 5. [Greatest lower bound of contracts]

Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ have a greatest lower bound $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ defined by:

$$\mathbf{A} = \mathbf{A}_1 \sqcup \mathbf{A}_2 \quad (40)$$

$$\mathbf{G} = ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) \quad (41)$$

Lemma 6. [Least upper bound of contracts]

Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ have a least upper bound $\mathbf{C} = (\mathbf{A}, \mathbf{G})$

defined by:

$$\mathbf{A} = \mathbf{A}_1 \sqcap \mathbf{A}_2 \tag{42}$$

$$\mathbf{G} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \tag{43}$$

Property 11. (\mathbb{C}, \preceq) is a distributive lattice of supremum $(\{\cup\}, \mathbb{P}^*)$ and infimum $(\mathbb{P}^*, \{\cup\})$.

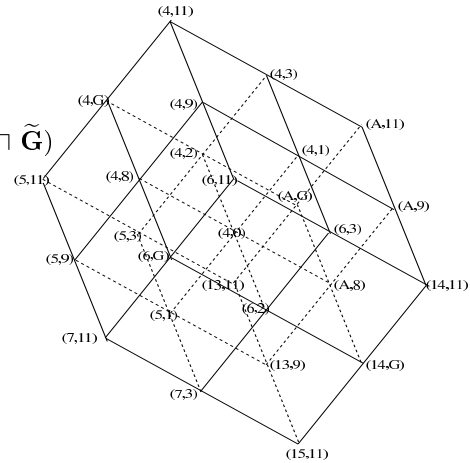
Property 12. A contract $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ has a complementary $\widetilde{\mathbf{C}}_1 = (\mathbf{A}_2, \mathbf{G}_2)$ iff $\mathbf{A}_1 = \widetilde{\mathbf{G}}_2$; this complementary is then $\widetilde{\mathbf{C}}_1 = (\mathbf{G}_1, \widetilde{\mathbf{A}}_1)$.

Definition 17. [Variable elimination in contract] Let x a variable, $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ a contract, the elimination of x in \mathbf{C} is the contract $\mathbf{C}_{\setminus x}$ defined by:
 $\mathbf{C}_{\setminus x} =_{\Delta} (\mathbf{A}_{|\forall x}, \mathbf{G}_{|\exists x})$

Property 13. A contract \mathbf{C} refines the elimination of a variable in \mathbf{C} : $\mathbf{C} \preceq \mathbf{C}_{\setminus x}$

The lattice of contracts *filtering equivalent* to (\mathbf{A}, \mathbf{G}) forms a cube presented using the following notations for filters:

- | | |
|--|--|
| 0 $\{\cup\}$ | 8 $\mathbf{A} \sqcap \mathbf{G}$ |
| 1 $\widetilde{\mathbf{A}} \sqcap \widetilde{\mathbf{G}}$ | 9 $(\mathbf{A} \sqcap \mathbf{G}) \sqcup (\widetilde{\mathbf{A}} \sqcap \widetilde{\mathbf{G}})$ |
| 2 $\widetilde{\mathbf{A}} \sqcap \mathbf{G}$ | 10 \mathbf{G} |
| 3 $\widetilde{\mathbf{A}}$ | 11 $\widetilde{\mathbf{A}} \sqcup \mathbf{G}$ |
| 4 $\mathbf{A} \sqcap \widetilde{\mathbf{G}}$ | 12 \mathbf{A} |
| 5 $\widetilde{\mathbf{G}}$ | 13 $\mathbf{A} \sqcup \widetilde{\mathbf{G}}$ |
| 6 $(\mathbf{A} \sqcap \widetilde{\mathbf{G}}) \sqcup (\widetilde{\mathbf{A}} \sqcap \mathbf{G})$ | 14 $\mathbf{A} \sqcup \mathbf{G}$ |
| 7 $\widetilde{\mathbf{A}} \sqcup \widetilde{\mathbf{G}}$ | 15 \mathbb{P}^* |



5 Related work

The use of contracts has been advocated for a long time in computer science [2, 4] and, more recently, has been successfully applied in object-oriented software engineering [3]. In object-oriented programming, a contract is characterized by a pair of *assumption* and *guarantee*. The *assumption* specifies hypothesis which has to be satisfied by the component in order to provide the *guarantee*.

In the context of software engineering, the notion of contract has been adapted for a wide variety of languages and formalisms, but the central notion of time needed for reactive system design is not always taken into account.

As an example, some extensions of OCL with linear or branching-time temporal logics have been proposed in [8, 11], focusing on the expressivity of the proposed constraint language (the way constraints may talk about the internals of classes and objects), and considering a fixed “sequence of states”. This is a serious limitation for concurrent system design, as this sequence becomes an interleaving of that of individual objects.

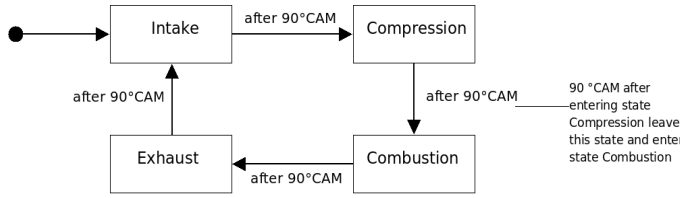
In the theory of interface automata [7], the notion of interface offers benefits similar to our notion of contracts and for the purpose of formal verification (checking interface compatibility). In that context, it is indeed irrelevant to separate the assumptions from guarantees. This becomes of importance in a more general-purpose software engineering context, because separation allows more flexibility in finding (contra-variant) compatibility relations between components.

In [1], a system of contracts with similar aims of genericity is proposed. By contrast to our domain-theoretical approach, the Speeds project considers an automata-based approach, which is indeed dual but makes notions such as the complementary of a contract more difficult to express from within the model. Also, the proposed approach chooses to leave the role of variables in contracts unspecified, thereby missing algebraic relations such as inclusion (as found in our model).

In [6], a notion of synchronous contracts has additionally been proposed for the programming language LUSTRE. In this approach, contracts are executable specifications timely paced by a clock (the clock of the component itself). This yields an approach which can hardly achieve compositionally as the clock of a composition of contracts needs to be the same as that of its constituting components or be explicitly related to them by sampling relations.

6 Discussion

We illustrate the distinctive features of our contract algebra by considering the specification of a four-stroke engine and its translation into observers in the synchronous language Signal.



The figure represents a state machine that denotes the successive operation modes of a 4-stroke engine : *Intake*, *Compression*, *Combustion*, and *Exhaust*. They are driven by the camshaft whose position is measured in angle degrees.

The angle of the camshaft defines a discrete timing reference, the *clock cam*, measured in degrees CAM° , of initial value 0. Transitions in the state machine are triggered by measures of the camshaft angle. The variables *cam*, *Intake*, *Combustion*, *Compression*, *Exhaust* model the behavior of the engine. We wish to define a contract to stipulate that intake always takes place in the first quarter on the camshaft revolution. To do this, we define the process-filter of the assumption. It should be a measure of the environmental variable *cam*. Namely *cam* should be in the first quarter. Under these assumptions, the state machine should be guaranteed to be in the intake mode, hence the process-filter for the guarantee.

$$\mathbf{A}_{Intake} = cam \text{ modulo } 360^\circ < 90 \quad \mathbf{G}_{Intake} = Intake$$

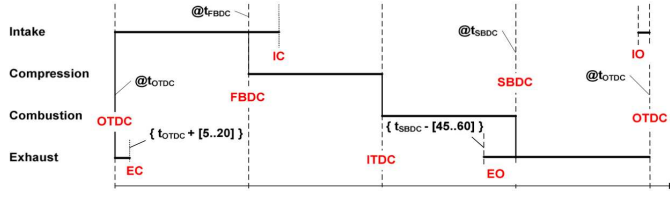
A beneficial feature of our algebra is that the separation of environmental assumptions and system guarantees is facilitated by the unpaired possibility to naturally express the complementary of a process-filter.

Had we used (interface) automata to model \mathbf{A}_{Intake} , it would have been (in general) much more challenging to define the engine not being in the intake mode just as it is to define the complementary of an automaton. Here, it is simply defined by $\widetilde{\mathbf{A}_{Intake}} = cam \text{ modulo } 360^\circ \geq 90$.

Furthermore, the generic structure of processes in contracts finds a direct instance and compositional translation into the synchronous multi-clocked model of computation of Signal [9].

$$A_{intake} = true \text{ when } (cam \text{ modulo } 360 < 90) \quad G_{intake} = true \text{ when } intake \text{ default } false$$

A subtlety of the Signal language is that the contract not only talks about the value, true or false, of the signals, but also about the status of the signal names, present or absent. Hence, the signal A_{intake} is present and true iff *cam* is present and less than 90. Hence, in Signal, the complementary of the assumptions is simply defined by $\widetilde{\mathbf{A}_{Intake}} = false \text{ when } A_{intake} \text{ default } true$ to mean that it is true iff *cam* is absent or bigger than 90. Notice that the clock (or reference in time) of $\widetilde{\mathbf{A}_{Intake}}$ need not be explicitly related to or ordered with \mathbf{A}_{Intake} or \mathbf{G}_{Intake} : it implicitly and partially relates to the *cam* clock. Had we used a strictly synchronous model of computation, as in [6], it would have been more difficult to compositionally define the complementary of a proposition without altering the clock of the environment itself or explicitly rating the clock of the assumption and guarantee to that of the environment (the *camshatf*).



Beside its Boolean structure, which allows for logical reasoning and normalization of contracts, our algebra supports the capability to compositionally refine contracts. For instance, consider a more precise model of the 4-stroke engine found in [5], left.

To additionally require that, while in the intake mode, the engine should reach the EC state between 6 and 20 degrees, one will simply compose the intake contract with the additional filter.

$$A_{EC} = \text{true when } (5 < \text{cam modulo } 360 < 21) \quad G_{EC} = \text{true when } EC \text{ default false}$$

7 Conclusion

Starting from the choice of an abstract characterization of behaviors as functions from variables to a domain of values (Booleans, integers, series, sets of tagged values, continuous functions), we introduced the notion of process-filters to formally characterize the logical device that filters behaviors from process much like the assumption and guarantee of a contract do. In our model, a process p fulfils its requirements (or satisfies) (\mathbf{A}, \mathbf{G}) if either it is rejected by \mathbf{A} (it is then out of the scope of the contract (\mathbf{A}, \mathbf{G})), or it is accepted by \mathbf{G} .

Our main result is that the structure of process-filters is a Boolean algebra. This rich structure allows for reasoning on contracts with great flexibility to abstract, refine and combine them. In addition to that, and unlike the related work, the negation of a contract can formally be expressed from within the model. Moreover, contracts are not limited to expressing safety properties, as is the case in most related frameworks, but encompass the expression of liveness properties. This is all again due to the central notion of process-filter.

We can observe that a given contract can be expressed with one single process-filter that filters the same set of processes. The filtering equivalence relation is satisfied if we consider the normalized form $(\mathbb{P}^*, \mathbf{G} \sqcup \tilde{\mathbf{A}})$ (since we have $\forall (\mathbf{A}, \mathbf{G}) \in \mathbb{C}, (\mathbf{A}, \mathbf{G}) \rightsquigarrow (\mathbb{P}^*, \mathbf{G} \sqcup \tilde{\mathbf{A}})$). In this case, the process-filter $\mathbf{G} \sqcup \tilde{\mathbf{A}}$ is sufficient to express a given property.

The manipulation of contracts has indeed a software engineering aspect. In particular, it might be considered that the process-filter \mathbf{A} defines the properties that the execution environment of the component must satisfy so as the guarantees represented by \mathbf{G} be satisfied by the component. Hypotheses on the environment can be expressed by properties on the input variables of the component, then the process-filter \mathbf{A} controls a set of variables included in the input variables of the component. While \mathbf{G} controls a set of variables included in the

input/output variables of the component. Then, the interpretation of a contract (\mathbf{A}, \mathbf{G}) is: if the environment of the component satisfies \mathbf{A} , then the component satisfies \mathbf{G} .

More generally, the process-filter \mathbf{G} might be considered as an abstraction of accepted processes and the process-filter \mathbf{A} as being an abstraction of the possible process-filters \mathbf{G} .

In the aim of assessing the generality and scalability of our approach, we are presently designing a module system based on the paradigm of contract for Signal and applying it to the specification of a component-based design process. The paradigm we are putting forward is to regard a contract as the behavioral type of a module or component and to use it for the elaboration of the functional architecture of a system together with a proof obligation that validates the correctness of assumptions and guarantees made while constructing that architecture.

References

- [1] Benveniste A., Caillaud B., and Passerone R. A generic model of contracts for embedded systems. In *Research report N°6214, INRIA-IRISA, S4 Team*, 2007.
- [2] Martin A. and Lamport L. Composing specifications. In *ACM Trans. ProgramLang. Syst. 15*, pages 73–132, 1993.
- [3] Meyer B. *Object-Oriented Software Construction, Second Edition*. ISE Inc., Santa Barbara, 1997.
- [4] Hoare C.A.R. An axiomatic basis for computer programing. In *Communications of the ACM*, pages 576–583, 1969.
- [5] André Charles, Mallet Frédéric, and Peraldi-Frati Marie-Agnès. A multi-form time approach to real-time system modeling. Research Report RR 2007-14-FR, I3S, 2007.
- [6] Maraninchi F. and Morel L. Logical-time contracts for reactive embedded components. In *In 30th EUROMICRO Conference on Component-Based Software Engineering Track, ECBSE'04, Rennes, France*, 2004.
- [7] Alfaro L. and Henzinger T. A. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120, 2001.
- [8] Ziemann P. and Gogolla M. An extension of OCL with temporal logic. In *Critical Systems Development with UML-Proceedings of the UML'02 workshop, Technische Universität München, Institut für Informatik*, pages 53–62, 2002.
- [9] Le Guernic Paul, Talpin Jean-Pierre, and Le Lann Jean-Christophe. Polychrony for system design. *Journal for Circuits, Systems and Computers*, Special Issue on Application Specific Hardware Design, 2003.
- [10] Abramsky S. and Jung A. Domain theory. In *Handbook of logic in computer science (vol. 3): semantic structures*, pages 1–168. Oxford University Press, 1997.
- [11] Flake S. and Mueller W. An OCL extension for realtime constraints. In *Lecture Notes in Computer Science 2263*, pages 150–171, 2001.

A Proofs of Section 2

Property 1. When \mathbf{W} , \mathbf{X} , \mathbf{Y} , \mathbf{Z} are finite sets of variables, \mathbf{Y} , \mathbf{Z} nonempty, p , q strict processes:

$$\begin{aligned} \text{var}(p) \subseteq \mathbf{Z} \subseteq \mathbf{Y} &\implies (p^{\mathbf{Z}|\mathbf{Y}} = p^{\mathbf{Y}}) \wedge (p^{\mathbf{Y}}|_{\mathbf{Z}} = p^{\mathbf{Z}}) \\ \text{var}(p) = \text{var}(q) \subseteq \mathbf{Y} &\implies (((p \cap q)^{\mathbf{Y}} = (p^{\mathbf{Y}} \cap q^{\mathbf{Y}})) \wedge ((p \cup q)^{\mathbf{Y}} = (p^{\mathbf{Y}} \cup q^{\mathbf{Y}}))) \\ \text{var}(p) = \text{var}(q) \subseteq \mathbf{Y} &\implies ((p \subseteq q) \iff (p^{\mathbf{Y}} \subseteq q^{\mathbf{Y}})) \\ \mathbf{X} \subseteq \text{var}(p) = \text{var}(q) &\implies ((p \subseteq q) \implies (p|_{\mathbf{X}} \subseteq q|_{\mathbf{X}})) \end{aligned}$$

Proof. proofs are immediate from equation 3 and equation 4

Property 2. (\mathbb{P}, \preceq) is a poset.

Proof. proof is immediate from equation 4 and property 2

Property 3. The complementary \tilde{p} of a strict process p is reduced iff p is reduced; \tilde{p} and p control the same set of variables $\text{var}(p)$.

Proof. is immediate considering the definitions of complementary (definition 3), variable control (definition 6) and reduced process (definition 7).

Property 4. The upper set of a strict process p contains a unique process $p^{\mathbf{Y}}$ defined on a given set of variables $\mathbf{Y} \supseteq \text{var}(p)$; the process p and its extension $p^{\mathbf{Y}}$ control the same set of variables, that is the set of variables controlled by the reduction of p .

$$((q \in [\preceq \uparrow p]) \wedge (r \in [\preceq \uparrow p]) \wedge (\text{var}(q) = \text{var}(r))) \implies (q = r) \text{equation 12}$$

$$(\text{var}(p) \subseteq \mathbf{Y}) \implies ((p^{\mathbf{Y}}) \triangleright \text{var}(\tilde{p})) \text{equation 13}$$

$$((\text{var}(p) \cup \text{var}(q)) \subseteq \mathbf{Y}) \implies ((p^{\mathbf{Y}} = q^{\mathbf{Y}}) \implies (\tilde{p} = \tilde{q})) \text{equation 14}$$

Proof. **equation 12:**

$$\begin{aligned} ((q \in [\preceq \uparrow p]) \wedge (r \in [\preceq \uparrow p])) &\iff ((p \preceq q) \wedge (p \preceq r)) \quad (\text{equation 9}) \\ &\iff ((p^{\text{var}(q)} = q) \wedge (p^{\text{var}(r)} = r)) \quad (\text{definition 5}) \\ &\implies ((\text{var}(q) = \text{var}(r)) \implies (q = r)) \end{aligned}$$

Proof. **equation 13:**

$$\begin{aligned}
& (\forall x \in \text{var}(\bar{p})) (p \triangleright x); \text{ let } q = p^{\mathbf{Y}} \\
(p \triangleright x) & \implies (x \in \text{var}(p)), (x \in \text{var}(p)) \implies (x \in \mathbf{Y}) \quad (\text{equation 10}) \\
& \implies (\exists a \in (p|_{(\mathbf{X} \setminus \{x\})})) \quad (\text{equation 10}) \\
& \quad (\exists b \in p) (b|_{(\mathbf{X} \setminus \{x\})} = a) \\
& \quad \text{and } (\exists v \in \mathcal{D}) (\forall b \in p) ((b|_{(\mathbf{X} \setminus \{x\})} = a) \implies (b(x) \neq v)) \\
& \implies (\exists a \in (p|_{(\mathbf{X} \setminus \{x\})})) \\
& \quad (\exists e \in q) (e|_{(\mathbf{X} \setminus \{x\})} = a) \\
& \quad \text{and } (\exists v \in \mathcal{D}) (\forall e \in q) ((e|_{(\mathbf{X} \setminus \{x\})} = a) \implies (e(x) \neq v)) \\
& \implies (\exists a \in (p|_{(\mathbf{X} \setminus \{x\})})) (\exists d \in (q|_{(\mathbf{Y} \setminus \{x\})})) \\
& \quad (\exists e \in q) (e|_{(\mathbf{Y} \setminus \{x\})} = d) \wedge (d|_{(\mathbf{X} \setminus \{x\})} = (e|_{(\mathbf{X} \setminus \{x\})} = a)) \\
& \quad \text{and } (\exists v \in \mathcal{D}) (\forall e \in q) ((e|_{(\mathbf{Y} \setminus \{x\})} = d) \implies (e(x) \neq v)) \\
& \implies (\exists d \in (q|_{(\mathbf{Y} \setminus \{x\})})) \\
& \quad (\exists e \in p) (e|_{(\mathbf{Y} \setminus \{x\})} = d) \\
& \quad \text{and } (\exists v \in \mathcal{D}) (\forall e \in p) ((e|_{(\mathbf{Y} \setminus \{x\})} = d) \implies (e(x) \neq v)) \\
& \implies (q \triangleright x) \quad (\text{equation 10}) \square
\end{aligned}$$

Proof. equation 14:

From $(p^{\mathbf{Y}} = q^{\mathbf{Y}})$ and equation 13 we get that p and q control $\text{var}(\bar{p})$ and $\text{var}(\bar{q})$; thus p and q control $\text{var}(\bar{p}) \cup \text{var}(\bar{q})$; this implies that $\text{var}(\bar{p}) = \text{var}(\bar{q})$ and then $\bar{p} = \bar{q}$

Property 5.

$$[[\bar{\Upsilon} \uparrow \bar{\Omega}]_{\underline{C}}] = \{\mathcal{U}\} \quad (\text{equation 16})$$

$$[[\bar{\Upsilon} \uparrow \bar{\Omega}]_{\underline{C}}] = \mathbb{P}^* \quad (\text{equation 17})$$

Proof. equation 16:

$$[[\bar{\Upsilon} \uparrow \bar{\Omega}]_{\underline{C}}] = [\{\mathcal{U}\}]_{\underline{C}} = \{p \in \mathbb{P}^* / p \subseteq \emptyset\} = \{\mathcal{U}\} \square$$

Proof. equation 17:

$$\begin{aligned}
[[\bar{\Upsilon} \uparrow \bar{\Omega}]_{\underline{C}}] &= [(\{\mathbb{B}_{\mathbf{X}}\}_{\mathbf{X} \subset \mathcal{C}, \mathcal{V}})]_{\underline{C}} \\
&= \{p \in \mathbb{P}^* / (\exists \mathbf{X}, \emptyset \subsetneq \mathbf{X} \subset \mathcal{C}, \mathcal{V}) (p \subseteq \mathbb{B}_{\mathbf{X}})\} \\
&= \mathbb{P}^* \square
\end{aligned}$$

B Proofs of Section 3

Property 6. The variable set of a process p , that belongs to a process-filter generated by a reduced process \bar{r} , contains the variable set of this process \bar{r} . The generator of a process-filter is unique; we refer to it as \mathbf{R} . Finally Ω generates the set of all processes (including \mathcal{U}), \mathcal{U} belongs to all filters. Formally ($\forall p, r, s \in \mathbb{P}^*$):

$$\begin{aligned}
 (p \in [\bar{r}]) &\implies (var(\bar{r}) \subseteq var(p)) \text{ (equation 18)} \\
 [\bar{r}] = [\bar{s}] &\iff \bar{r} = \bar{s} \text{ (equation 19)} \\
 \Omega \in [\bar{r}] &\iff [\bar{r}] = \mathbb{P}^* \text{ (equation 20)} \\
 \mathcal{U} \in \mathbf{R} &\text{ equation 21} \text{nonumber}
 \end{aligned} \tag{44}$$

Proof. equation 18:

from definitions given by equation 9 and equation 15, p is included in some extension of \bar{r} \square

Proof. equation 19: (\Leftarrow is obvious)

$$\begin{aligned}
 [\bar{r}] = [\bar{s}] &\implies (\bar{r} \in [\bar{s}]) \wedge (\bar{s} \in [\bar{r}]) \text{ (equation 9, equation 15)} \\
 &\implies (var(\bar{s}) \subseteq var(\bar{r})) \wedge (var(\bar{r}) \subseteq var(\bar{s})) \text{ (equation 18)} \\
 &\implies (\bar{r} \subseteq \bar{s}) \wedge (\bar{s} \subseteq \bar{r}) \text{ (equation 9, equation 15)} \square
 \end{aligned}$$

Proof. equation 20: (\Leftarrow is obvious)

$$\begin{aligned}
 \Omega \in [\bar{r}] &\implies (var(\bar{r}) \subseteq var(\Omega) = \emptyset) \text{ (equation 18)} \\
 &\implies \bar{r} = \Omega \square
 \end{aligned}$$

Proof. equation 21:

Direct consequence of process-filter definition 8 \square

Theorem 1. A strict process p belongs to a process-filter \mathbf{R} iff

$$(\forall \mathbf{X}, \mathbf{Y} \subseteq_l \mathcal{V}) (var(\mathbf{R}) \subseteq \mathbf{X} \subseteq \mathbf{Y}), (p \in \mathbf{R}) \iff \begin{cases} (var(\mathbf{R}) \subseteq var(p)) \\ (var(p) \subseteq \mathbf{Y}) \implies ((p|_{\mathbf{Y}})|_{\mathbf{X}} \subseteq \mathbf{R}^{\bar{X}}) \end{cases}$$

Proof. (\implies)

$$(p \in \mathbf{R}) \implies (\text{var}(\mathbf{R}) \subseteq \text{var}(p)) \quad (\text{equation 22 and equation 18})$$

$$(p \in \mathbf{R}) \iff (\exists s \in \mathbf{R})((p \subseteq s) \wedge (s \in [\overset{\nabla}{\neq} \uparrow \mathbf{R}])) \quad (\text{definition 8 and equation 15})$$

$$(p \in \mathbf{R}) \iff (\exists s \in \mathbf{R})((p \subseteq s) \wedge (s = \mathbf{R}^{\overset{\nabla}{|}\text{var}(p)})) \quad (\text{equation 9})$$

$$(p \in \mathbf{R}) \iff (p \subseteq \mathbf{R}^{\overset{\nabla}{|}\text{var}(p)})$$

It results from corollary 1 (equation 7 and equation 8) that:

$$(p \in \mathbf{R}) \implies (((\text{var}(p) \subseteq \mathbf{Y}) \wedge (\mathbf{X} \subseteq \mathbf{Y})) \implies ((p^{\overset{\nabla}{|}\mathbf{Y}})_{|\mathbf{X}} \subseteq ((\mathbf{R}^{\overset{\nabla}{|}\text{var}(p)})^{\overset{\nabla}{|}\mathbf{Y}})_{|\mathbf{X}}))$$

$$(p \in \mathbf{R}) \implies ((p^{\overset{\nabla}{|}\mathbf{Y}})_{|\mathbf{X}} \subseteq (\mathbf{R}^{\overset{\nabla}{|}\mathbf{Y}})_{|\mathbf{X}} = (\mathbf{R}^{\overset{\nabla}{|}\mathbf{X}})) \quad (\text{corollary 1-equation 5}) \quad \square$$

Proof. (\impliedby)

$$\begin{aligned} ((p^{\overset{\nabla}{|}\mathbf{Y}})_{|\mathbf{X}} \subseteq (\mathbf{R}^{\overset{\nabla}{|}\mathbf{X}})) &\implies (p_{|\text{var}(\mathbf{R})} \subseteq \mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{R})}) \quad \text{hyphothesis with } (\mathbf{Y} = \text{var}(p), \mathbf{X} = \text{var}(\mathbf{R})) \\ &\implies (p \subseteq \mathbf{R}^{\overset{\nabla}{|}\text{var}(p)}) \quad (\text{corollary 1}) \\ &\implies p \in \mathbf{R} \quad \square \end{aligned}$$

Corollary 1. The two equivalent properties are satisfied:

$$\mathbf{R} \subseteq \mathbf{S} \iff ((\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge (\mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \subseteq \mathbf{S}^{\overset{\nabla}{|}\text{var}(\mathbf{S})})) \quad (\text{equation 23})$$

$$\mathbf{R} \subseteq \mathbf{S} \iff \mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \in \mathbf{S} \quad (\text{equation 24})$$

Proof.

is an application of theorem 1:

$$\implies \text{since } \mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \in \mathbf{S} \iff (\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge (\mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \subseteq \mathbf{S}^{\overset{\nabla}{|}\text{var}(\mathbf{S})}) \quad (\text{theorem 1})$$

$$\text{and } \mathbf{R} \subseteq \mathbf{S} \implies \mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \in \mathbf{S} \quad (\mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \in \mathbf{R})$$

$$\text{we have } \mathbf{R} \subseteq \mathbf{S} \implies (\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge (\mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \subseteq \mathbf{S}^{\overset{\nabla}{|}\text{var}(\mathbf{S})})$$

$$\impliedby \text{since } (\forall p \in \mathbb{P})(p \in \mathbf{R} \iff (\text{var}(\mathbf{R}) \subseteq \text{var}(p)) \wedge (p_{|\text{var}(\mathbf{R})} \subseteq \mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{R})})) \quad (\text{theorem 1})$$

$$\text{and } (\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge (\mathbf{R}^{\overset{\nabla}{|}\text{var}(\mathbf{S})} \subseteq \mathbf{S}^{\overset{\nabla}{|}\text{var}(\mathbf{S})}) \quad (\text{hypothesis})$$

$$\text{we get } (\forall p \in \mathbb{P})(p \in \mathbf{R} \implies (\text{var}(\mathbf{S}) \subseteq \text{var}(p)) \wedge (p_{|\text{var}(\mathbf{S})} \subseteq \mathbf{S}^{\overset{\nabla}{|}\text{var}(\mathbf{S})}))$$

$$\text{then } (\forall p \in \mathbb{P})(p \in \mathbf{R} \implies p \in \mathbf{S})$$

Property 7. Strict process-filters \mathbf{R} and \mathbf{S} satisfy $(\mathbf{R} \subseteq \mathbf{S}) \iff ((\text{var}(\mathbf{S}) \subseteq \text{var}(\mathbf{R})) \wedge \mathbf{R} \sqsubseteq \mathbf{S})$

Proof. direct application of corollary 1 and definition 9

Property 8: (Φ, \sqsubseteq) is a poset.

Proof. the relation \sqsubseteq is clearly reflexive and antisymmetric; transitivity is easily shown using corollary 1-equation 5

Lemma 1. (Φ, \sqsubseteq) is a lattice with \mathbb{P}^* as supremum and $\{\emptyset\}$ as infimum; the infimum (or conjunction) $\mathbf{R} \sqcap \mathbf{S}$, the supremum (or disjunction) $\mathbf{R} \sqcup \mathbf{S}$ are defined by:

$$\begin{aligned} \{\emptyset\} \sqcap \mathbf{R} &= \mathbf{R} \sqcap \{\emptyset\} = \{\emptyset\} \\ (\mathbf{R} \neq \{\emptyset\} \wedge \mathbf{S} \neq \{\emptyset\}) &\implies \mathbf{R} \sqcap \mathbf{S} =_{\Delta} [[\overset{\nabla}{\neq} \uparrow \overset{\nabla}{p}] \downarrow_{\sqsubseteq}] \text{ (equation 29)} \\ &\text{where } p = (\overset{\nabla}{\mathbf{R}} \cap \overset{\nabla}{\mathbf{S}}), \mathbf{V} = \text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}) \\ \{\emptyset\} \sqcup \mathbf{R} &= \mathbf{R} \sqcup \{\emptyset\} = \mathbf{R} \\ (\mathbf{R} \neq \{\emptyset\} \wedge \mathbf{S} \neq \{\emptyset\}) &\implies \mathbf{R} \sqcup \mathbf{S} =_{\Delta} [[\overset{\nabla}{\neq} \uparrow \overset{\nabla}{p}] \downarrow_{\sqsubseteq}] \text{ (equation 31)} \\ &\text{where } p = (\overset{\nabla}{\mathbf{R}} \cup \overset{\nabla}{\mathbf{S}}), \mathbf{V} = \text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}) \end{aligned}$$

Proof.

let $\mathbf{W}_1 = (\text{var}(\mathbf{R}) \cup \text{var}(\mathbf{T}))$, $\mathbf{W}_2 = (\text{var}(\mathbf{S}) \cup \text{var}(\mathbf{T}))$, $\mathbf{W} = (\mathbf{W}_1 \cup \mathbf{W}_2)$,

- $\mathbf{R} \sqcap \mathbf{S}$, as defined above, is the infimum of \mathbf{R} and \mathbf{S} (we ignore the obvious case where $(\mathbf{R} = \{\emptyset\} \vee \mathbf{S} = \{\emptyset\})$)

$$((\mathbf{T} \sqsubseteq \mathbf{R}) \wedge (\mathbf{T} \sqsubseteq \mathbf{S})) \iff ((\overset{\nabla}{\mathbf{T}} \subseteq \overset{\nabla}{\mathbf{R}}) \wedge (\overset{\nabla}{\mathbf{T}} \subseteq \overset{\nabla}{\mathbf{S}}))$$

using theorem 1 we get

$$((\mathbf{T} \sqsubseteq \mathbf{R}) \wedge (\mathbf{T} \sqsubseteq \mathbf{S})) \iff ((\overset{\nabla}{\mathbf{T}} \subseteq \overset{\nabla}{\mathbf{R}}) \wedge (\overset{\nabla}{\mathbf{T}} \subseteq \overset{\nabla}{\mathbf{S}}))$$

$$((\mathbf{T} \sqsubseteq \mathbf{R}) \wedge (\mathbf{T} \sqsubseteq \mathbf{S})) \iff (\overset{\nabla}{\mathbf{T}} \subseteq \overset{\nabla}{\mathbf{R}} \cap \overset{\nabla}{\mathbf{S}})$$

thus $((\mathbf{T} \sqsubseteq \mathbf{R}) \wedge (\mathbf{T} \sqsubseteq \mathbf{S}))$ implies $(\mathbf{T} \sqsubseteq (\mathbf{R} \sqcap \mathbf{S}))$; taking $\mathbf{T} = (\mathbf{R} \sqcap \mathbf{S})$ we get $((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{R}) \wedge ((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{S}) \square$

- $\mathbf{R} \sqcup \mathbf{S}$, as defined above, is the supremum of \mathbf{R} and \mathbf{S} (we ignore the obvious case where $(\mathbf{R} = \{\emptyset\} \vee \mathbf{S} = \{\emptyset\})$)

$$((\mathbf{R} \sqsubseteq \mathbf{T}) \wedge (\mathbf{S} \sqsubseteq \mathbf{T})) \iff ((\overset{\nabla}{\mathbf{R}} \subseteq \overset{\nabla}{\mathbf{T}}) \wedge (\overset{\nabla}{\mathbf{S}} \subseteq \overset{\nabla}{\mathbf{T}}))$$

using theorem 1 we get

$$((\mathbf{R} \sqsubseteq \mathbf{T}) \wedge (\mathbf{S} \sqsubseteq \mathbf{T})) \iff ((\overset{\nabla}{\mathbf{R}} \subseteq \overset{\nabla}{\mathbf{T}}) \wedge (\overset{\nabla}{\mathbf{S}} \subseteq \overset{\nabla}{\mathbf{T}}))$$

$((\mathbf{R} \sqsubseteq \mathbf{T}) \wedge (\mathbf{S} \sqsubseteq \mathbf{T})) \iff (\mathbf{R} \sqcup \mathbf{S} \sqsubseteq \mathbf{T})$
 thus $((\mathbf{R} \sqsubseteq \mathbf{T}) \wedge (\mathbf{S} \sqsubseteq \mathbf{T}))$ implies $((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T})$; taking $\mathbf{T} = (\mathbf{R} \sqcup \mathbf{S})$ we get $((\mathbf{R} \sqsubseteq (\mathbf{R} \sqcup \mathbf{S})) \wedge (\mathbf{S} \sqsubseteq (\mathbf{R} \sqcup \mathbf{S}))) \square$

Lemma 2. The following properties hold: for $\mathbf{R}, \mathbf{S}, \mathbf{T}$ strict process-filters,

$$((\mathbf{R} \cap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \subseteq (\text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T})) \text{ (equation 32)}$$

and thus $((\mathbf{R} \cap \mathbf{S}) \sqsubseteq (\mathbf{R} \sqcap \mathbf{S}))$ (take $(\mathbf{R} \sqcap \mathbf{S}) = \mathbf{T}$) (equation 33)

$$((\mathbf{R} \cup \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \subseteq (\text{var}(\mathbf{R}) \cap \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T})) \text{ (equation 34)}$$

and thus $((\mathbf{R} \cup \mathbf{S}) \sqsubseteq (\mathbf{R} \sqcup \mathbf{S}))$ (take $(\mathbf{R} \sqcup \mathbf{S}) = \mathbf{T}$) (equation 35)

Proof.

let $\mathbf{X} = \text{var}(\mathbf{R}), \mathbf{Y} = \text{var}(\mathbf{S}), \mathbf{Z} = \text{var}(\mathbf{T}), \mathbf{V} = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$

+equation 32

$$((\mathbf{R} \cap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff (\forall p \in \mathbb{P}) (p \in (\mathbf{R} \cap \mathbf{S}) \implies (p \in \mathbf{T}))$$

let $\mathbf{W} = \text{var}(p) \cup \mathbf{V}$

$$(p \in \mathbf{R}) \iff ((\mathbf{X} \subseteq \text{var}(p)) \wedge ((p|_{\mathbf{W}})^{\nabla|\mathbf{V}} \subseteq \mathbf{R}))$$

theorem 1 $(p \in \mathbf{S}) \iff ((\mathbf{Y} \subseteq \text{var}(p)) \wedge ((p|_{\mathbf{W}})^{\nabla|\mathbf{V}} \subseteq \mathbf{S}))$

$$(p \in \mathbf{T}) \iff ((\mathbf{Z} \subseteq \text{var}(p)) \wedge ((p|_{\mathbf{W}})^{\nabla|\mathbf{V}} \subseteq \mathbf{T}))$$

$$(p \in (\mathbf{R} \cap \mathbf{S})) \iff (((\mathbf{X} \cup \mathbf{Y}) \subseteq \text{var}(p)) \wedge ((p|_{\mathbf{W}})^{\nabla|\mathbf{V}} \subseteq (\mathbf{R} \cap \mathbf{S})))$$

$$((\mathbf{R} \cap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \subseteq \mathbf{V} = (\text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \cap \mathbf{S})^{\nabla|\mathbf{V}} \subseteq \mathbf{T}))$$

equation 29 $\mathbf{R} \sqcap \mathbf{S} = [[\uparrow \mathbf{R} \mathbf{S}] \downarrow_{\subseteq}]$ where $\mathbf{R} \mathbf{S} = (\mathbf{R} \cap \mathbf{S})^{\nabla|\mathbf{X} \cup \mathbf{Y}}$

definition 9 $\mathbf{R} \sqcap \mathbf{S} \sqsubseteq \mathbf{T} \iff \mathbf{R} \mathbf{S}^{\nabla|\mathbf{W}} \subseteq \mathbf{T}^{\nabla|\mathbf{W}}$ where $\mathbf{W} = \text{var}(\mathbf{R} \mathbf{S}) \cup \mathbf{Z}$

$$((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff \mathbf{R} \mathbf{S}^{\nabla|\mathbf{V}} \subseteq \mathbf{T}^{\nabla|\mathbf{V}} \text{ (we have } \text{var}(\mathbf{R} \sqcap \mathbf{S}) \subseteq (\mathbf{X} \cup \mathbf{Y}) \text{ and thus } \mathbf{W} \subseteq \mathbf{V})$$

$$((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff (\mathbf{R} \cap \mathbf{S})^{\nabla|\mathbf{X} \cup \mathbf{Y}} \subseteq \mathbf{T}^{\nabla|\mathbf{V}}$$

$$((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff (\mathbf{R} \cap \mathbf{S})^{\nabla|\mathbf{V}} \subseteq \mathbf{T}^{\nabla|\mathbf{V}} \text{ equation 6 and equation 5}$$

finally

$$((\mathbf{R} \cap \mathbf{S}) \sqsubseteq \mathbf{T}) \iff ((\text{var}(\mathbf{T}) \subseteq (\text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcap \mathbf{S}) \sqsubseteq \mathbf{T}))$$

$$\begin{aligned}
& \text{+equation 34} \\
((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}) & \iff (\forall p \in \mathbb{P}) (p \in (\mathbf{R} \cup \mathbf{S}) \implies (p \in \mathbf{T})) \\
& \text{with similar reasons, one get} \\
((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}) & \iff ((\text{var}(\mathbf{T}) \subseteq (\text{var}(\mathbf{R}) \cap \text{var}(\mathbf{S}))) \wedge ((\overset{\nabla}{\mathbf{R}} \overset{\nabla}{\cup} \overset{\nabla}{\mathbf{S}}) \subseteq \overset{\nabla}{\mathbf{T}})) \\
((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}) & \iff (\overset{\nabla}{\mathbf{R}} \overset{\nabla}{\cup} \overset{\nabla}{\mathbf{S}}) \subseteq \overset{\nabla}{\mathbf{T}} \quad \text{equation 6 and equation 5} \\
& \text{finally} \\
((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}) & \iff ((\text{var}(\mathbf{T}) \subseteq (\text{var}(\mathbf{R}) \cap \text{var}(\mathbf{S}))) \wedge ((\mathbf{R} \sqcup \mathbf{S}) \sqsubseteq \mathbf{T}))
\end{aligned}$$

Theorem 2: (Φ, \sqsubseteq) is a Boolean algebra with \mathbb{P}^* as 1, $\{\emptyset\}$ as 0 and the complementary $\tilde{\mathbf{R}}$.
Proof.

since (Φ, \sqsubseteq) is a lattice, we know that:

- $\mathbf{R} \sqcup (\mathbf{S} \sqcap \mathbf{T}) = (\mathbf{R} \sqcup \mathbf{S}) \sqcup \mathbf{T}$ (and its dual)
- $\mathbf{R} \sqcup \mathbf{S} = \mathbf{S} \sqcup \mathbf{R}$ (and its dual)
- $\mathbf{R} \sqcup (\mathbf{R} \sqcap \mathbf{S}) = \mathbf{R}$ (and its dual)

thus we only have to prove:

- $\tilde{\mathbf{R}} \sqcap \mathbf{R} = \{\emptyset\}$; this is a direct consequence of the definition:
either $\mathbf{R} = \{\emptyset\}$ and $\tilde{\mathbf{R}} \sqcap \{\emptyset\} = \{\emptyset\}$ (see \sqcap equation 29)
or $\mathbf{R} \neq \{\emptyset\}$; in this case $\tilde{\mathbf{R}} \sqcap \mathbf{R}$ is empty (see \sqcap definition in lemma 1): $[[\overset{\nabla}{\mathbf{R}} \uparrow \tilde{\emptyset}] \downarrow \sqsubseteq] = \{\emptyset\}$
- $\tilde{\mathbf{R}} \sqcup \mathbf{R} = \mathbb{P}^*$; this is a direct consequence of the definition:
either $\mathbf{R} = \{\emptyset\}, \tilde{\{\emptyset\}} = \mathbb{P}^*$ (see $\tilde{\mathbf{R}}$ definition above) and $\tilde{\mathbf{R}} \sqcup \{\emptyset\} = \mathbf{R}$ (see \sqcup equation 31)
or $\mathbf{R} \neq \{\emptyset\}$; in this case $\tilde{\mathbf{R}} \sqcup \mathbf{R}$ is $\mathbb{B}_{\text{var}(\mathbf{R})}$ whose reduction is Ω : $[[\overset{\nabla}{\mathbf{R}} \uparrow \tilde{\Omega}] \downarrow \sqsubseteq] = \mathbb{P}^*$
- $\mathbf{R} \sqcup (\mathbf{S} \sqcap \mathbf{T}) = (\mathbf{R} \sqcup \mathbf{S}) \sqcap (\mathbf{R} \sqcup \mathbf{T})$ (or its dual)
If $\mathbf{R} = \{\emptyset\}$ we have $\mathbf{R} \sqcup (\mathbf{S} \sqcap \mathbf{T}) = (\mathbf{S} \sqcap \mathbf{T})$, $(\mathbf{R} \sqcup \mathbf{S}) = \mathbf{S}$, $(\mathbf{R} \sqcup \mathbf{T}) = \mathbf{T}$
If $\mathbf{S} = \{\emptyset\}$ (or commutatively $\mathbf{T} = \{\emptyset\}$) we have $(\mathbf{S} \sqcap \mathbf{T}) = \{\emptyset\}$, then $\mathbf{R} \sqcup (\mathbf{S} \sqcap \mathbf{T}) = \mathbf{R}$; on the other hand we get $(\mathbf{R} \sqcup \mathbf{S}) \sqcap (\mathbf{R} \sqcup \mathbf{T}) = \mathbf{R} \sqcap (\mathbf{R} \sqcup \mathbf{T})$; (Φ, \sqsubseteq) being a lattice, $\mathbf{R} \sqcap (\mathbf{R} \sqcup \mathbf{T}) = \mathbf{R}$
If none of $\mathbf{R}, \mathbf{S}, \mathbf{T}$ is equal to $\{\emptyset\}$, from definitions and theorem 1 it comes that
 $\mathbf{R} \sqcup (\mathbf{S} \sqcap \mathbf{T}) = (\mathbf{R} \sqcup \mathbf{S}) \sqcap (\mathbf{R} \sqcup \mathbf{T})$ iff
 $(\overset{\nabla}{\mathbf{R}}) \overset{\nabla}{\cup} ((\overset{\nabla}{\mathbf{S}}) \overset{\nabla}{\cap} (\overset{\nabla}{\mathbf{T}})) = ((\overset{\nabla}{\mathbf{R}}) \overset{\nabla}{\cup} (\overset{\nabla}{\mathbf{S}})) \overset{\nabla}{\cap} ((\overset{\nabla}{\mathbf{R}}) \overset{\nabla}{\cup} (\overset{\nabla}{\mathbf{T}}))$
(where $\mathbf{V} = \text{var}(\mathbf{R}) \cup \text{var}(\mathbf{S}) \cup \text{var}(\mathbf{T})$); \square

property 9: $\mathbf{R}_{|\forall x} \sqsubseteq \mathbf{R} \sqsubseteq \mathbf{R}_{|\exists x}$

Proof. by definition, the generator of $\mathbf{R}_{|\exists x}$ is a restriction of the generator of \mathbf{R} , and then (theorem 1) all processes in \mathbf{R} belong to $\mathbf{R}_{|\exists x}$. Thus $\widetilde{\mathbf{R}} \sqsubseteq \widetilde{\mathbf{R}}_{|\exists x}$ and finally (theorem 2) $\widetilde{\widetilde{\mathbf{R}}}_{|\exists x} \sqsubseteq \widetilde{\mathbf{R}} = \mathbf{R}$

C Proofs of Section 4

Corollary 4. $p \models \mathbf{C} \iff [\widehat{p}] \sqsubseteq (\widetilde{A} \sqcup \mathbf{G})$

Proof. Boolean algebra property:

$$([\widehat{p}] \sqcap \mathbf{A}) \sqsubseteq \mathbf{G} \iff (([\widehat{p}] \sqcap \mathbf{A}) \sqcap \widetilde{\mathbf{G}} = \{\emptyset\}) \iff ([\widehat{p}] \sqsubseteq (\widetilde{A} \sqcup \mathbf{G})) \quad \square$$

Lemma 3. This relation satisfies the following property:

$$(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2) \iff (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqsubseteq (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)$$

Proof.

We have (corollary 4) $(\mathbf{A}_1, \mathbf{G}_1) \rightsquigarrow (\mathbf{A}_2, \mathbf{G}_2)$

$$\iff (\forall p \in \mathbb{P}) (([\widehat{p}] \sqsubseteq (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1)) \implies ([\widehat{p}] \sqsubseteq (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$$

\implies take p equal to the generator of $(\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1)$

$$\iff (\forall p \in \mathbb{P}) (([\widehat{p}] \sqsubseteq (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1)) \implies ([\widehat{p}] \sqsubseteq (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqsubseteq (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$$

\square

Corollary 5. Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ are *filtering equivalent* if and only if $(\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) = (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)$.

Proof. from corollary 4 \square

Lemma 4. $(\mathbf{A}_1, \mathbf{G}_1) \preceq (\mathbf{A}_2, \mathbf{G}_2)$ iff the three following properties are satisfied: (a) $\mathbf{A}_2 \sqsubseteq \mathbf{A}_1$ (b) $(\mathbf{A}_2 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_2$ and (c) $\mathbf{G}_1 \sqsubseteq \mathbf{A}_1 \sqcup \mathbf{G}_2$.

Proof.

Item (i) in definition 16 is equivalent to $(\mathbf{A}_2 \sqsubseteq (\mathbf{A}_1 \sqcup \mathbf{G}_2)) \wedge ((\mathbf{A}_2 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_2)$ (lemma 3 and Boolean algebra properties); then (i) and (a) is equivalent to (a) and (b). \square

Property 10. (\mathbb{C}, \preceq) is a poset.

Proof. \preceq is clearly reflexive; let us prove transitivity and antisymetry;

- Transitivity:

(a) $(\mathbf{A}_3 \sqsubseteq \mathbf{A}_1)$: from (a) in lemma 4 we get $\mathbf{A}_3 \sqsubseteq \mathbf{A}_2 \sqsubseteq \mathbf{A}_1$

(b) $((\mathbf{A}_3 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_3)$:
 $((\mathbf{A}_2 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_2) \implies ((\mathbf{A}_3 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_1) \sqsubseteq (\mathbf{A}_3 \sqcap \mathbf{G}_2))$ (boolean lattice)
 $\implies ((\mathbf{A}_3 \sqcap \mathbf{G}_1) \sqsubseteq ((\mathbf{A}_3 \sqcap \mathbf{G}_2) \sqsubseteq \mathbf{G}_3))$
 $(\mathbf{A}_3 \sqsubseteq \mathbf{A}_2$ and (b) in lemma 4)

(c) $(\mathbf{G}_1 \sqsubseteq \mathbf{A}_1 \sqcup \mathbf{G}_3)$:
 $\mathbf{G}_1 \sqsubseteq (\mathbf{A}_1 \sqcup \mathbf{G}_2) \sqsubseteq (\mathbf{A}_1 \sqcup (\mathbf{A}_2 \sqcup \mathbf{G}_3)) = (\mathbf{A}_1 \sqcup \mathbf{G}_3)$

- Antisymmetry: from (a) in lemma 4 we get $\mathbf{A}_1 = \mathbf{A}_2$; applying this equality and rules of Boolean algebra to (b) and (c) in lemma 4 we get

$$\begin{aligned} ((\mathbf{G}_1 \sqsubseteq (\mathbf{A}_1 \sqcup \mathbf{G}_2)) \wedge (\mathbf{G}_2 \sqsubseteq (\mathbf{A}_1 \sqcup \mathbf{G}_1))) &\implies ((\mathbf{A}_1 \sqcup \mathbf{G}_1) = (\mathbf{A}_1 \sqcup \mathbf{G}_2)) \\ ((\mathbf{A}_1 \sqcap \mathbf{G}_2) \sqsubseteq \mathbf{G}_1) \wedge ((\mathbf{A}_1 \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{G}_2) &\implies ((\mathbf{A}_1 \sqcap \mathbf{G}_1) = (\mathbf{A}_1 \sqcap \mathbf{G}_2)) \\ &\implies (\mathbf{G}_1 = \mathbf{G}_2) \end{aligned}$$

and then $\mathbf{C}_1 = \mathbf{C}_2$

□

Lemma 5-a. A contract $\mathbf{D} = (\mathbf{B}, \mathbf{H})$ is a lower bound of two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ iff it satisfies the following property:

$$\mathbf{B} = \mathbf{A}_1 \sqcup \mathbf{A}_2 \sqcup \mathbf{B} \tag{45}$$

$$\begin{aligned} \mathbf{H} = \mathbf{H} \sqcap ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2) \\ \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2))) \end{aligned} \tag{46}$$

Proof. $(\mathbf{D} \preceq \mathbf{C}_1) \wedge (\mathbf{D} \preceq \mathbf{C}_2) \iff$ (lemma 4)

$$\begin{aligned} \mathbf{A}_1 \sqsubseteq \mathbf{B} &\quad \wedge \quad \mathbf{A}_2 \sqsubseteq \mathbf{B} \\ ((\mathbf{H} \sqcap \mathbf{A}_1) \sqsubseteq \mathbf{G}_1) &\quad \wedge \quad ((\mathbf{H} \sqcap \mathbf{A}_2) \sqsubseteq \mathbf{G}_2) \\ \mathbf{H} \sqsubseteq (\mathbf{B} \sqcup \mathbf{G}_1) &\quad \wedge \quad \mathbf{H} \sqsubseteq (\mathbf{B} \sqcup \mathbf{G}_2) \\ &\iff \text{(lattice properties and Boolean algebra rules)} \end{aligned}$$

$$\begin{aligned} \mathbf{B} &= \mathbf{A}_1 \sqcup \mathbf{A}_2 \sqcup \mathbf{B} \\ \mathbf{H} &= \mathbf{H} \sqcap ((\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) \end{aligned}$$

\iff (using first relation in last one)

$$\begin{aligned} \mathbf{B} &= \mathbf{A}_1 \sqcup \mathbf{A}_2 \sqcup \mathbf{B} \\ \mathbf{H} &= \mathbf{H} \sqcap (((\mathbf{A}_1 \sqcup \mathbf{A}_2 \sqcup \mathbf{B}) \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) \end{aligned}$$

finally Boolean algebra rules gives:

$$\begin{aligned} \mathbf{H} &= \mathbf{H} \sqcap ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) \\ &\quad \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)) \end{aligned}$$

□

Lemma 6-a. A contract $\mathbf{D} = (\mathbf{B}, \mathbf{H})$ is an upper bound of two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ iff it satisfies the following property:

$$\mathbf{B} = \mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{B} \quad (47)$$

$$\mathbf{H} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2) \sqcap \mathbf{B}) \sqcup \mathbf{H} \quad (48)$$

Proof. $(\mathbf{C}_1 \preceq \mathbf{D}) \wedge (\mathbf{C}_2 \preceq \mathbf{D}) \iff$ (lemma 4)

$$\begin{aligned} \mathbf{B} \sqsubseteq \mathbf{A}_1 &\quad \wedge \quad \mathbf{B} \sqsubseteq \mathbf{A}_2 \\ (\mathbf{B} \sqcap \mathbf{G}_1) \sqsubseteq \mathbf{H} &\quad \wedge \quad (\mathbf{B} \sqcap \mathbf{G}_2) \sqsubseteq \mathbf{H} \\ \mathbf{G}_1 \sqsubseteq \mathbf{A}_1 \sqcup \mathbf{H} &\quad \wedge \quad \mathbf{G}_2 \sqsubseteq \mathbf{A}_2 \sqcup \mathbf{H} \\ &\iff \text{(lattice properties and Boolean algebra rules)} \end{aligned}$$

$$\begin{aligned} \mathbf{B} &= \mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{B} \\ \mathbf{H} &= (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{B} \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2)) \sqcup \mathbf{H} \end{aligned}$$

\iff (using first relation in second one)

$$\begin{aligned} \mathbf{B} &= \mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{B} \\ \mathbf{H} &= (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{B} \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2)) \sqcup \mathbf{H} \end{aligned}$$

□

Lemma 5. [Greatest lower bound of contracts]

Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ have a greatest lower bound $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ defined by:

$$\mathbf{A} = \mathbf{A}_1 \sqcup \mathbf{A}_2 \quad (\text{equation 40})$$

$$\mathbf{G} = ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) \quad (\text{equation 41})$$

Proof.

- $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ is a lower bound:
 - \mathbf{C} is a lower bound iff (lemma 5-a)
 - $\mathbf{A} = \mathbf{A} \sqcup \mathbf{A}_1 \sqcup \mathbf{A}_2$ (clearly satisfied by equation 40)
 - $\mathbf{G} = \mathbf{G} \sqcap ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$

As $\mathbf{G} = ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2))$ (equation 41) we get by substitution: \mathbf{C} is a lower bound iff

$$\mathbf{G} = \mathbf{G} \sqcap (\mathbf{G} \sqcup (\mathbf{A} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$$

□

- if $\mathbf{D} = (\mathbf{B}, \mathbf{H})$ is a lower bound of \mathbf{C}_1 and \mathbf{C}_2 then \mathbf{D} refines \mathbf{C} :
 \mathbf{D} is a lower bound of \mathbf{C}_1 and \mathbf{C}_2 iff (lemma C)
 $\mathbf{B} = \mathbf{A}_1 \sqcup \mathbf{A}_2 \sqcup \mathbf{B}$
 $\mathbf{H} = \mathbf{H} \sqcap ((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2) \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$
 \mathbf{D} is a lower bound of \mathbf{C}_1 and \mathbf{C}_2 iff (substitute \mathbf{A} to its value in right hand side of first relation and \mathbf{G} to its value in right hand side of second one)
 $\mathbf{B} = \mathbf{A} \sqcup \mathbf{B}$
 $\mathbf{H} = \mathbf{H} \sqcap (\mathbf{G} \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$
 \mathbf{D} refines \mathbf{C} iff (lemma 4)

(a) $\mathbf{A} \sqsubseteq \mathbf{B}$ (this property is satisfied)

(b) $(\mathbf{A} \sqcap \mathbf{H}) \sqsubseteq \mathbf{G}$. We have

$$\begin{aligned} (\mathbf{A} \sqcap \mathbf{H}) &= \mathbf{A} \sqcap (\mathbf{G} \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2))) \\ &\quad (\text{since } \mathbf{A} \sqsubseteq \mathbf{B}) \end{aligned}$$

$$\begin{aligned} (\mathbf{A} \sqcap \mathbf{H}) &= \mathbf{A} \sqcap (\mathbf{G} \sqcup (\mathbf{A} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2))) \\ &\quad (\text{since } (\mathbf{A} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)) \sqsubseteq \mathbf{G}) \end{aligned}$$

$$(\mathbf{A} \sqcap \mathbf{H}) = \mathbf{A} \sqcap \mathbf{G} \sqsubseteq \mathbf{G}$$

□

(c) $\mathbf{H} \sqsubseteq \mathbf{B} \sqcup \mathbf{G}$. We have

$$\mathbf{H} \sqsubseteq (\mathbf{G} \sqcup (\mathbf{B} \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \mathbf{G}_1) \sqcap (\widetilde{\mathbf{A}}_2 \sqcup \mathbf{G}_2)))$$

□

Lemma 6. [Least upper bound of contracts]

Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ have a least upper bound $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ defined by:

$$\mathbf{A} = \mathbf{A}_1 \sqcap \mathbf{A}_2 \quad (\text{equation 42})$$

$$\mathbf{G} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \quad (\text{equation 43})$$

Proof.

- $\mathbf{C} = (\mathbf{A}, \mathbf{G})$ is an upper bound:
 - \mathbf{C} is an upper bound iff (lemma 6-a)
 - $\mathbf{A} = \mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{A}$, clearly satisfied
 - $\mathbf{G} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2) \sqcap \mathbf{A}) \sqcup \mathbf{G}$
 - We have $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2) \sqcap \mathbf{A}) \sqsubseteq \mathbf{G}$ \square
 - if $\mathbf{D} = (\mathbf{B}, \mathbf{H})$ is an upper bound of \mathbf{C}_1 and \mathbf{C}_2 then \mathbf{C} refines \mathbf{D}
 - \mathbf{D} is an upper bound of \mathbf{C}_1 and \mathbf{C}_2 iff (lemma C)
 - $\mathbf{B} = \mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{B}$
 - $\mathbf{H} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2) \sqcap \mathbf{B}) \sqcup \mathbf{H}$
 - \mathbf{C} refines \mathbf{D} iff lemma 4
- (a) $\mathbf{B} \sqsubseteq \mathbf{A}$ (this property is satisfied)
- (b) $(\mathbf{B} \sqcap \mathbf{G}) \sqsubseteq \mathbf{H}$. We have
 $\mathbf{H} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G} \sqcap \mathbf{B}) \sqcup \mathbf{H}$
- (c) $\mathbf{G} \sqsubseteq \mathbf{A} \sqcup \mathbf{H}$. We have
 $\mathbf{A} \sqcup \mathbf{H} = \mathbf{A} \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup \mathbf{H}$
 and $\mathbf{G} \sqsubseteq \mathbf{A} \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2)$

□

Property 11. (\mathbb{C}, \preceq) is a distributive lattice of supremum $(\{\mathcal{U}\}, \mathbb{P}^*)$ and infimum $(\mathbb{P}^*, \{\mathcal{U}\})$.

Proof.

- $(\{\mathcal{U}\}, \mathbb{P}^*)$ is the supremum: for all contracts (\mathbf{A}, \mathbf{G}) $((\mathbf{A}, \mathbf{G}) \sqsubseteq (\{\mathcal{U}\}, \mathbb{P}^*))$; trivially checked from lemma 4
- $(\mathbb{P}^*, \{\mathcal{U}\})$ is the infimum: for all contracts (\mathbf{A}, \mathbf{G}) $(\mathbb{P}^*, \{\mathcal{U}\}) \sqsubseteq (\mathbf{A}, \mathbf{G})$ trivially checked from lemma 4
- Distributivity: $(\mathbf{C}_1 \uparrow (\mathbf{C}_2 \downarrow \mathbf{C}_3)) = ((\mathbf{C}_1 \uparrow \mathbf{C}_2) \downarrow (\mathbf{C}_1 \uparrow \mathbf{C}_3))$; let $\mathbf{C}_{23} = (\mathbf{A}_{23}, \mathbf{G}_{23}) = (\mathbf{C}_2 \downarrow \mathbf{C}_3)$, where (lemma 5)
 - $\mathbf{A}_{23} = \mathbf{A}_2 \sqcup \mathbf{A}_3$
 - $\mathbf{G}_{23} = ((\mathbf{A}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_2) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3) \sqcup (\mathbf{G}_2 \sqcap \mathbf{G}_3))$
 - $\mathbf{C}_{12} = (\mathbf{A}_{12}, \mathbf{G}_{12}) = (\mathbf{C}_1 \uparrow \mathbf{C}_2)$, where (lemma 6)
 - $\mathbf{A}_{12} = \mathbf{A}_1 \sqcap \mathbf{A}_2$
 - $\mathbf{G}_{12} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2)$
 - $\mathbf{C}_{13} = (\mathbf{A}_{13}, \mathbf{G}_{13}) = (\mathbf{C}_1 \uparrow \mathbf{C}_3)$, where (lemma 6)
 - $\mathbf{A}_{13} = \mathbf{A}_1 \sqcap \mathbf{A}_3$
 - $\mathbf{G}_{13} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \sqcup (\mathbf{A}_3 \sqcap \mathbf{G}_1) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_3)$
 - $\mathbf{C}_{1(23)} = (\mathbf{A}_{1(23)}, \mathbf{G}_{1(23)}) = (\mathbf{C}_1 \uparrow \mathbf{C}_{23})$ and
 - $\mathbf{C}_{(12)(13)} = (\mathbf{A}_{(12)(13)}, \mathbf{G}_{(12)(13)}) = (\mathbf{C}_{12} \downarrow \mathbf{C}_{13})$

$$\begin{aligned}
& - \mathbf{A}_{1(23)} = \mathbf{A}_{(12)(13)} \\
& \text{from equation 40, equation 42 and theorem 2 one get} \\
& \mathbf{A}_{1(23)} = \mathbf{A}_1 \sqcap (\mathbf{A}_2 \sqcup \mathbf{A}_3) = (\mathbf{A}_1 \sqcap \mathbf{A}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_3) = \mathbf{A}_{(12)(13)} \\
& - \mathbf{G}_{1(23)} = \mathbf{G}_{(12)(13)} \\
& \mathbf{G}_{1(23)} \text{ is defined by lemma 6:} \\
& \mathbf{G}_{1(23)} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_{23} \sqcap \mathbf{G}_{23}) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_{23}) \sqcup (\mathbf{A}_{23} \sqcap \mathbf{G}_1) \\
& \text{Variable substitution:} \\
& \mathbf{G}_{1(23)} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \widetilde{\mathbf{A}}_3) \\
& \quad \sqcap ((\mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3) \sqcup (\mathbf{G}_2 \sqcap \mathbf{G}_3)) \\
& \quad \sqcup (\mathbf{A}_1) \\
& \quad \sqcap ((\mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3) \sqcup (\mathbf{G}_2 \sqcap \mathbf{G}_3)) \\
& \quad \sqcup ((\mathbf{A}_2 \sqcup \mathbf{A}_3) \\
& \quad \sqcap \mathbf{G}_1) \\
& \text{Distributivity} \\
& \mathbf{G}_{1(23)} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \text{Factorization by } (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1), (\widetilde{\mathbf{A}}_1 \sqcap \widetilde{\mathbf{G}}_1), (\mathbf{A}_1 \sqcap \widetilde{\mathbf{G}}_1), (\mathbf{A}_1 \sqcap \mathbf{G}_1), \\
& \mathbf{G}_{1(23)} = (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \widetilde{\mathbf{G}}_1 \sqcap \mathbf{G}_2 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \widetilde{\mathbf{G}}_1) \\
& \quad \sqcap ((\mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{G}_2 \sqcap \mathbf{G}_3)) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcap ((\mathbf{G}_2 \sqcap \mathbf{G}_3) \\
& \quad \sqcup \mathbf{A}_2 \\
& \quad \sqcup \mathbf{A}_3)) \\
& \mathbf{G}_{(12)(13)} \text{ is defined by lemma 5:} \\
& \mathbf{G}_{(12)(13)} = (\mathbf{A}_{12} \sqcap \widetilde{\mathbf{A}}_{13} \sqcap \mathbf{G}_{12}) \sqcup (\widetilde{\mathbf{A}}_{12} \sqcap \mathbf{A}_{13} \sqcap \mathbf{G}_{13}) \sqcup (\mathbf{G}_{12} \sqcap \mathbf{G}_{13}) \\
& \text{Variable substitution:} \\
& \mathbf{G}_{(12)(13)} = (\mathbf{A}_1 \sqcap \mathbf{A}_2) \\
& \quad \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \widetilde{\mathbf{A}}_3) \\
& \quad \sqcap ((\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1)
\end{aligned}$$

$$\begin{aligned}
& \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \\
& \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \\
& \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2)) \\
& \sqcup ((\widetilde{\mathbf{A}}_1 \sqcup \widetilde{\mathbf{A}}_2) \\
& \quad \sqcap \mathbf{A}_1 \sqcap \mathbf{A}_3 \\
& \quad \sqcap ((\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_3))) \\
& \sqcup (((\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \\
& \quad \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2)) \\
& \quad \sqcap ((\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \\
& \quad \sqcup (\mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \quad \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_3)))
\end{aligned}$$

Distributivity

$$\begin{aligned}
\mathbf{G}_{(12)(13)} &= (\mathbf{A}_1 \sqcap \mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \\
& \quad \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_2)) \\
& \sqcup (\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \\
& \quad \sqcap (\mathbf{G}_1 \sqcup \mathbf{G}_3)) \\
& \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \\
& \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2 \sqcap \mathbf{A}_1 \sqcap \mathbf{G}_3) \\
& \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \\
& \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1 \sqcap \mathbf{A}_1 \sqcap \mathbf{G}_3) \\
& \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_3) \\
& \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_1) \\
& \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2 \sqcap (\mathbf{A}_1 \sqcap \mathbf{G}_3))
\end{aligned}$$

Factorization by $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1)$, $(\widetilde{\mathbf{A}}_1 \sqcap \widetilde{\mathbf{G}}_1)$, $(\mathbf{A}_1 \sqcap \widetilde{\mathbf{G}}_1)$, $(\mathbf{A}_1 \sqcap \mathbf{G}_1)$,

$$\begin{aligned}
\mathbf{G}_{(12)(13)} &= (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \\
& \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \widetilde{\mathbf{G}}_1 \sqcap \mathbf{G}_2 \sqcap \mathbf{G}_3) \\
& \sqcup (\mathbf{A}_1 \sqcap \widetilde{\mathbf{G}}_1 \\
& \quad \sqcap ((\mathbf{A}_2 \sqcap \widetilde{\mathbf{A}}_3 \sqcap \mathbf{G}_2) \\
& \quad \sqcup \widetilde{\mathbf{A}}_2 \sqcap \mathbf{A}_3 \sqcap \mathbf{G}_3 \\
& \quad \sqcup (\mathbf{G}_2 \sqcap \mathbf{G}_3))) \\
& \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_1)
\end{aligned}$$

$$\begin{array}{l} \cdot \\ \cdot \\ \cdot \end{array} \quad \begin{array}{l} \sqcap ((\mathbf{G}_2 \sqcap \mathbf{G}_3) \\ \sqcup \mathbf{A}_2 \\ \sqcup \mathbf{A}_3)) \end{array}$$

□

Property 12. A contract $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ has a complementary $\widetilde{\mathbf{C}}_1 = (\mathbf{A}_2, \mathbf{G}_2)$ iff $\mathbf{A}_1 = \widetilde{\mathbf{G}}_1$; this complementary is then $\widetilde{\mathbf{C}}_1 = (\mathbf{G}_1, \widetilde{\mathbf{G}}_1)$

Proof.

Two contracts $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ are complementary iff their greatest lower bound is the contract $(\mathbb{P}^*, \{\mathcal{U}\})$, and their least upper bound is the contract $(\{\mathcal{U}\}, \mathbb{P}^*)$. Thus $\mathbf{C}_1 = (\mathbf{A}_1, \mathbf{G}_1)$ and $\mathbf{C}_2 = (\mathbf{A}_2, \mathbf{G}_2)$ are complementary iff they satisfy the following properties:

- 1 $\mathbf{A}_1 \sqcup \mathbf{A}_2 = \mathbb{P}^*$
- 2 $((\mathbf{A}_1 \sqcap \widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{A}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{G}_1 \sqcap \mathbf{G}_2)) = \{\mathcal{U}\}$
- 3 $\mathbf{A}_1 \sqcap \mathbf{A}_2 = \{\mathcal{U}\}$
- 4 $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\widetilde{\mathbf{A}}_2 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2) \sqcup (\mathbf{A}_2 \sqcap \mathbf{G}_1) = \mathbb{P}^*$

these properties are satisfied iff

- 13 $\mathbf{A}_2 = \widetilde{\mathbf{A}}_1$ (equations 1 and 3)
- 2-1 $(\mathbf{A}_1 \sqcap \mathbf{G}_1) = \{\mathcal{U}\}$
- 2-2 $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_2) = \{\mathcal{U}\}$
- 2-3 $(\mathbf{G}_1 \sqcap \mathbf{G}_2) = \{\mathcal{U}\}$
- 4 $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_1) \sqcup (\mathbf{A}_1 \sqcap \mathbf{G}_2) = \mathbb{P}^*$

these properties are satisfied iff

- 13 $\mathbf{A}_2 = \widetilde{\mathbf{A}}_1$ (equations 1 and 3)
- 2-1 $(\mathbf{A}_1 \sqcap \mathbf{G}_1) = \{\mathcal{U}\}$
- 2-2 $(\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_2) = \{\mathcal{U}\}$
- 2-3 $(\mathbf{G}_1 \sqcap \mathbf{G}_2) = \{\mathcal{U}\}$
- 4 $(\mathbf{A}_1 \sqcup \widetilde{\mathbf{G}}_1) \sqcap (\widetilde{\mathbf{A}}_1 \sqcup \widetilde{\mathbf{G}}_2) = \{\mathcal{U}\}$

these properties are satisfied iff

$$13 \mathbf{A}_2 = \widetilde{\mathbf{A}}_1 \text{ (equations 1 and 3)}$$

$$2-1 (\mathbf{A}_1 \sqcap \mathbf{G}_1) = \{\top\}$$

$$2-2 (\widetilde{\mathbf{A}}_1 \sqcap \mathbf{G}_2) = \{\top\}$$

$$2-3 (\mathbf{G}_1 \sqcap \mathbf{G}_2) = \{\top\}$$

$$4-2 \mathbf{A}_1 \sqcap \widetilde{\mathbf{G}}_2 = \{\top\}$$

$$4-3 \widetilde{\mathbf{A}}_1 \sqcap \widetilde{\mathbf{G}}_1 = \{\top\}$$

$$4-4 \widetilde{\mathbf{G}}_1 \sqcap \widetilde{\mathbf{G}}_2 = \{\top\}$$

these properties are satisfied iff

- $\mathbf{A}_2 = \widetilde{\mathbf{A}}_1$ (equations 1 and 3)
- $\mathbf{G}_1 = \widetilde{\mathbf{A}}_1$ (equations 2-1 and 4-3)
- $\mathbf{G}_2 = \mathbf{A}_1$ (equations 2-2 and 4-2)

□

Property 13. A contract \mathbf{C} refines the elimination of a variable in \mathbf{C} : $\mathbf{C} \preceq \mathbf{C}_{\setminus x}$

Proof.

let $\mathbf{C} = (\mathbf{A}, \mathbf{G})$; from lemma 4 and definition 17, $\mathbf{C} \preceq \mathbf{C}_{\setminus x}$ iff the following properties are satisfied

- (a) $\mathbf{A}_{|\forall x} \sqsubseteq \mathbf{A}$
- (b) $(\mathbf{A}_{|\forall x} \sqcap \mathbf{G}) \sqsubseteq \mathbf{G}_{|\exists x}$
- (c) $\mathbf{G} \sqsubseteq \mathbf{A} \sqcup \mathbf{G}_{|\exists x}$

The satisfaction of these properties is a trivial consequence of property 9 and theorem 2

□



Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399