Configuration Theories

Pietro Cenciarelli

University of Rome, “La Sapienza”
Department of Computer Science - Via Salaria 113, 00198 Roma.
cenciarelli@dsi.uniroma1.it

Abstract. A new framework for describing concurrent systems is presented. Rules for composing configurations of concurrent programs are represented by sequents $\Gamma \vdash_\rho \Delta$, where Γ and Δ are sequences of partially ordered sets (of events) and ρ is a matrix of monotone maps from the components of Γ to the components of Δ. Such a sequent expresses that whenever a configuration has certain specified subposets of events (Γ), then it extends to a configuration containing one of several specified subposets (Δ). The structural rules of Gentzen's sequent calculus are decorated by suitable operations on matrices, where *cut* corresponds to product. The calculus thus obtained is shown to be sound with respect to interpretation in *configuration structures* [GG90]. Completeness is proven for a restriction of the calculus to finite sequents. As a case study we axiomatise the *Java* memory model, and formally derive a non-trivial property of thread-memory interaction.

Keywords: semantics, concurrency, configuration structures, sequent calculus, Java.

1 Introduction

The Java language specification [GJS96] is very precise in describing how the events of a Java computation may depend on each other. For instance, it is required that, whenever a thread θ (a lightweight process) modifies the content of its *working memory* by assigning a value to an instance variable while holding a lock on some object, that value must be copied to the *main memory* before θ is allowed to release the lock [ibid. §17.6]. While it is relatively easy to write a denotational model of Java (say, as a Petri net or as an event structure [Cen00]), it is unclear whether such a model, a description of some large and complicated graph, would serve its purpose, e.g. to provide a usable mathematical framework for validating program logics or for proving, for example, that a process respects the above protocol on locks.

While writing an operational semantics of Java [CKRW98], the author realised that the rules of interaction that processes must obey could be conveniently formalised by using the same stuff of which models are made: posets of events. What a rule gives is a recipe for arranging events into legal configurations. From the work on Java the general idea originated of a context calculus where posets of events representing fragments of concurrent computation combine into larger fragments according to language-dependent rules given in the form of axioms.

In the present paper we propose an axiomatic framework for describing concurrent systems. It is a sequent calculus, where sequents are made of posets and monotone injections specifying how the posets are allowed or required to match.

Sections 2 and 3 describe syntax and semantics of sequents. In Section 4 we give the structural rules of the calculus and prove soundness with respect to interpretation in *configuration structures* [GG90]. Completeness is proven in Section 5 for a restriction of the calculus to *finite* posets. The Java memory model is axiomatised by means of finite posets in Section 6 where a non-trivial property of thread interaction is proven formally. Note that the Java memory model to which we refer [GJS96, §17] is now under revision by the Java Community Process in order to make it support common optimising techniques that are currently disallowed. But of course the point of Section 6 is not to study specific features of the Java language, but to see how configuration theories fare in real life.

Notation. Here are some adopted notational conventions. An $m \times n$ *matrix* ρ in a *set* S is a doubly indexed family of elements ρ_{ij} of S ($i = 1 \ldots m$ and $j : 1 \ldots n$). When either m or n are 0, an $n \times m$ matrix is the empty family. The Greek letters ρ, σ, τ are used as metavariables for two-dimensional matrices. We write ρ_i instead of ρ_{i1} when ρ has size $m \times 1$. Similarly when ρ has size $1 \times n$.

If ρ and σ are matrices of size $m \times n$ and $r \times n$ respectively, we write $\rho\,;\sigma$ for the $(m + r) \times n$ matrix obtained by "placing ρ above σ": the ij-component of $\rho\,;\sigma$ is ρ_{ij} for $i \leq m$, while it is $\sigma_{(i-m)j}$ when $i > m$. Similarly, if ρ and σ are of size $m \times n$ and $m \times r$, we write $\rho\,,\sigma$ for the $m \times (n + r)$ matrix obtained by "placing ρ to the left of σ": the ij-component of ρ, σ is ρ_{ij} for $j \leq n$, while it is $\sigma_{i(j-n)}$ when $j > n$.

We write function composition in diagrammatical order.

2 Poset Sequents

If A and B are partially ordered sets (posets), we write $p : A \rightarrowtail B$ for a monomorphism in the category of posets, that is, an injective function preserving the order of elements. In general, a momomorphism $p : A \rightarrowtail B$ in a category $\mathcal{C}$ is called *strong* when, for any commuting square $e\,v = u\,p$ in $\mathcal{C}$, where $e : C \to D$ is an epimorphism, there exists a unique diagonal $d : D \to A$ such that $v = d\,p$. Then, any strong mono which is epimorphic is an isomorphism. The strong monos p in the category of posets are exactly those which *reflect* the order, that is: $p(a) \leq p(b)$ implies $a \leq b$. By a simple argument, if A and B are *finite*, if p is as above and there exists a mono $B \rightarrowtail A$, then p must reflect the order. Moreover, p must be surjective (an epimorphism), and therefore it must be an isomorphism. This result is used in the proof of Proposition 11.

In general, we use Γ, Δ ... as metavariables for sequences of posets, A, B ... for individual posets, and a, b ... for elements of posets. However, when the components of a sequence $\mathit{\Gamma}$ are not introduced explicitly by an equation $\mathit{\Gamma} = A_1, \dots A_m$, we write $\mathit{\Gamma}_i$ for the i-th element of $\mathit{\Gamma}$. Concatenation of sequences $\mathit{\Gamma}$ and $\mathit{\Delta}$ is written $\mathit{\Gamma}, \mathit{\Delta}$. If $\mathit{\Gamma} = A_1, \dots A_m$ and $\mathit{\Delta} = B_1 \dots B_n$ are finite sequences of posets, we write $\rho : \mathit{\Gamma} \to \mathit{\Delta}$ to mean that ρ is an $m \times n$ matrix of monos $\rho_{ij} : A_i \rightarrowtail B_j$. An $m \times 1$ matrix $\mathit{\Gamma} \to D$ is called an *interpretation* of $\mathit{\Gamma}$ in D.

Definition 1 *A* poset sequent $\mathit{\Gamma} \vdash_\rho \mathit{\Delta}$ *(just* sequent *for short) consists of two finite sequences* $\mathit{\Gamma}$ *and* $\mathit{\Delta}$ *of posets and an* $m \times n$ *matrix* $\rho : \mathit{\Gamma} \to \mathit{\Delta}$ *of monos.*

The posets in a sequent are meant to represent fragments of a configuration of events. The intuitive meaning of a sequent $\mathit{\Gamma} \vdash_\rho \mathit{\Delta}$ is that whenever a single configuration interprets *all* components of $\mathit{\Gamma}$, the interpretation extends along ρ to *at least one* component of $\mathit{\Delta}$. Of course the $\mathit{\Delta}_i$ may include more events than are mentioned in $\mathit{\Gamma}$, thus specifying what is required to happen after (or must have happened before) a certain combination ($\mathit{\Gamma}$) of events.

Let $\mathit{\Gamma} \vdash_\rho \mathit{\Delta}$ and $\mathit{\Pi} \vdash_\sigma \mathit{\Delta}$ be sequents. It is easy to check that $\mathit{\Gamma}, \mathit{\Pi} \vdash_{\rho;\sigma} \mathit{\Delta}$ is a "well formed" sequent, that is: $(\rho;\sigma) : \mathit{\Gamma}, \mathit{\Pi} \to \mathit{\Delta}$. Similarly, if $\mathit{\Gamma} \vdash_\rho \mathit{\Delta}$ and $\mathit{\Gamma} \vdash_\sigma \mathit{\Pi}$ are sequents, so is $\mathit{\Gamma} \vdash_{\rho,\sigma} \mathit{\Delta}, \mathit{\Pi}$. Finally, if $\rho : \mathit{\Gamma} \to A$ and $\sigma : A \to \mathit{\Delta}$ are matrices of size $m \times 1$ and $1 \times n$ respectively, we can form their *product* $\rho\sigma : \mathit{\Gamma} \to \mathit{\Delta}$, of size $m \times n$, by using function composition to multiply components. Hence, if $\mathit{\Gamma} \vdash_\rho A$ and $A \vdash_\sigma \mathit{\Delta}$ are sequents, then so is $\mathit{\Gamma} \vdash_{\rho\sigma} \mathit{\Delta}$.

Example. Let $B, C \vdash_{f;g} A$ and $A \vdash_{u,v} D, E$ be sequents, where $f : B \rightarrowtail A$, $g : C \rightarrowtail A$, $u : A \rightarrowtail D$ and $v : A \rightarrowtail E$. Then, $B, C \vdash_\rho D, E$ is a sequent, where

$$\rho = \begin{bmatrix} f \\ g \end{bmatrix} [u \ v] = \begin{bmatrix} fu & fv \\ gu & gv \end{bmatrix},$$

with $fu : B \rightarrowtail D \dots$ and $gv : C \rightarrowtail E$ as required.

□

There is no general construction for multiplying two matrices $\mathit{\Gamma} \to \mathit{\Delta} \to \mathit{\Pi}$. Some notion of summation on morphisms of the form $\mathit{\Gamma}_i \rightarrowtail \mathit{\Pi}_j$ would be needed for that. However, in Section 5 we use the following construction for multiplying a matrix by a vector of row vectors. Let $\rho : \mathit{\Gamma} \to B_1, \dots B_n$ be an $m \times n$ matrix of monos, and let $\sigma : [\sigma^{(1)}, \dots \sigma^{(n)}]$ be a vector where each $\sigma^{(i)}$ is a $1 \times k_i$ matrix $B_i \to \mathit{\Pi}^{(i)}$. The product $\rho_{(\text{-})i}\sigma^{(i)} : \mathit{\Gamma} \to \mathit{\Pi}^{(i)}$ has size $m \times k_i$. Pasting all such matrices together horizontally we obtain a matrix $\rho \Diamond \sigma = (\rho_{(\text{-})1}\sigma^{(1)}, \dots \rho_{(\text{-})n}\sigma^{(n)})$ of size $m \times (k_1 + \cdots + k_n)$. The $\Diamond$ construction is thus described by the following formation rule:

$$[\Diamond] \quad \frac{\mathit{\Gamma} \vdash_\rho B_1, \dots B_n \qquad B_1 \vdash_{\sigma^{(1)}} \mathit{\Pi}^{(1)} \quad \dots \quad B_n \vdash_{\sigma^{(n)}} \mathit{\Pi}^{(n)}}{\mathit{\Gamma} \vdash_{\rho \Diamond \sigma} \mathit{\Pi}^{(1)}, \dots \mathit{\Pi}^{(n)}} \quad (\sigma = [\sigma^{(1)}, \dots \sigma^{(n)}])$$

Let L be a set of *labels* to be thought of as action names. An L-labelled sequent is a sequent $\mathit{\Gamma} \vdash_\rho \mathit{\Delta}$ where all components X of $\mathit{\Gamma}$ and $\mathit{\Delta}$ are labelled by a function $X \to L$ and all components of ρ respect the labelling.

3 Configuration Structures

Definition 2 [GP95] *A* configuration structure *is a pair* $(E, \mathcal{C})$ *where* E *is a set, whose elements are called* events, *and* $\mathcal{C}$ *is a collection of subsets of* E*, called* configurations.

The events of a configuration structure (or just *structure* for short) can be viewed as occurrences of the actions a concurrent system may perform, while a configuration models a consistent state of the system, represented as the set of events occurred during computation up to that point. We write just $\mathcal{C}$ for $(E, \mathcal{C})$ when no confusion arises.

Configuration structures originate from [Win82], where they were introduced as an alternative way to address *event structures* [NPW81] (in the form known later as *prime event structures with binary conflict*). In [Win87] several closure conditions on the set of configurations of a structure $\mathcal{C}$ were given in order to get a precise match with *general event structures* (generalising those of [NPW81]). The requirements were: *finiteness* (if an event belongs to a configuration C, then it also belongs to a finite subconfiguration of C), *coincidence-freeness* (if two distinct events belong to a configuration C, then there exists a subconfiguration of C containing exactly one of them), closure under *bounded unions* and *non-emptyness* of $\mathcal{C}$.

In the framework of (general) event structures, configurations (as well as the order on events) are defined in terms of other mathematical structure. In the present paper, and following [GG90], we find it convenient to take the notion of configuration as primitive and that of order as derived. To this effect we adopt here (and do so implicitly) all of the above requirements *except* for closure under bounded unions, which is not needed for the treatment.

Let C be a configuration of a structure $\mathcal{C}$. We write $Sub(C)$ for the set $\{D \in \mathcal{C} \mid D \subseteq C\}$ of subconfigurations of C. Then, we let $\leq_C$ denote the binary relation on C such that $b \leq_C a$ if and only if, for all $D \in Sub(C)$, $a \in D$ implies $b \in D$. The set $\{b \in C \mid b \leq_C a\}$ is denoted by $C \downarrow a$, and similarly for $C \uparrow a$.

Proposition 3 *The relation* $\leq_C$ *is a* partial order. *Moreover, for all* $a \in C$*, the set* $C \downarrow a$ *is finite.*

The antisymmetry of $\leq$ (we omit indices when no confusion arises) is an immediate consequence of coincidence-freeness while finiteness of $C \downarrow a$ follows from the finiteness property on configurations (the converse does not hold). We use this property (only) in Section 6, where a formal rule expressing the groundedness of configurations is introduced to prove a property of Java. By the proposition above, we treat configurations as posets. If A is a poset, we write $(E_A, \mathcal{C}_A)$ for the structure whose events are the elements of A and whose configurations are the downwards closed subsets of A. When a poset A is treated as a configuration structure, it is meant $(E_A, \mathcal{C}_A)$.

In general, the collection of partial orders $\leq_C$, $C \in \mathcal{C}$, defined as above does not represent the causality relation on $\mathcal{C}$ faithfully. In particular, it does not hold that $a \leq_D b$ implies $a \leq_C b$ for $D \in Sub(C)$ (while the converse holds by definition). Calling *conservative* a structure where the above implication does hold, it is easy to check the following:

Proposition 4 *A configuration structure $\mathcal{C}$ is conservative if and only if it is has* downwards-closed bounded intersections*: for all $C \in \mathcal{C}$, for all $D, F \in Sub(C)$, and for all $a \in D \cap F$, if $b \leq_D a$ then $b \in F$.*

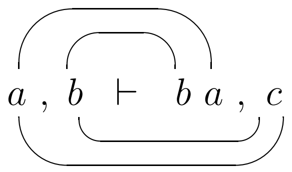
If C and D are configurations of a conservative structure, with $D \in Sub(C)$, the inclusion $D \subseteq C$ can be viewed as a morphism in the category of posets, which we write $(D, \leq_D) \hookrightarrow (C, \leq_C)$. We rely on conservativity in Definition 5, which is the heart of the present paper. Henceforth the configuration structures of discourse will implicitly be assumed conservative. Note that so are the *stable* structures of [GG01], which require closure under bounded intersections. These are precisely the structures where the order on a configuration determines its subconfigurations. Indeed all results in the present paper, which rely on weaker assumptions, specialise to stable structures.

Definition 5 *A structure $\mathcal{C}$ is said to* satisfy *a sequent $\Gamma \vdash_\rho \Delta$ when, for any configuration $C \in \mathcal{C}$ and interpretation $\pi : \Gamma \to C$, there exist a configuration $D \in \mathcal{C}$, a component $\Delta_k \in \Delta$ and a mono $q : \Delta_k \rightarrowtail D$ such that $C \in Sub(D)$ and, for all i, the following diagram commutes.*

$$\begin{tikzcd} \Gamma_i \arrow[r, "\rho_{ik}"] \arrow[d, "\pi_i"'] & \Delta_k \arrow[d, "q"] \\ C \arrow[r, hook] & D \end{tikzcd} \tag{1}$$

The notion of interpretation given above extends to labelled sequents and *labelled configuration structures* [GP95] in an obvious way. The reader should check that the above definition agrees with the intuitive meaning of sequents proposed in the previous section. Note that in the present setting we decided to attach no special *computational* meaning to the inclusions as $C \hookrightarrow D$ above. However, the notion of $\hookrightarrow$ must be strenthened in order to prove that satisfaction is preserved by *history preserving bisimulation* [GG01].

A sequent is called *valid* if it is satisfied by all structures. An example of valid sequent is $A \vdash_{id} A$. A slightly more complicated example is given in Diagram 2, which states that if a poset has an element a and it has an element b, then either a and b are the same element or they are distinct. The adopted graphical representation of sequents is to be read as follows: posets are separated by commas and no braces are used to hold elements of a set together. Vertical lines represent the order within each poset (where a below b means $a < b$), while their absence means no order. Links spanning across the turnstile represent the matrix of monos, where $a \frown b$ means $a \mapsto b$ by the corresponding matrix component.



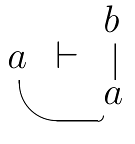
(2)

By the above conventions, (2) stands for a sequent $A, B \vdash_\rho C, D$ where $A = \{a\}$, $B = \{b\}$, $C = \{a, b\}$ (with a and b unordered) and $D = \{c\}$. Moreover, $\rho_{11}(a) = a$, $\rho_{12}(a) = c$ and so on.

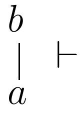
Example 6 $A, A \vdash_{id;id} A$ is *not* satisfied by the structure $A + A$, where $+$ denotes disjoint union. In fact, the two copies of A on the left side of the sequent are disjoint in the interpretation $A \xrightarrow{inl} A + A \xleftarrow{inr} A$, while the components of $(id; id)$ do overlap.

$\square$

Here are other examples. The sequent $\vdash$ denotes *absurdity*. Note that this sequent features empty sequences as antecedent and succedent and it is meant as decorated by the empty matrix. A structure satisfying $\vdash A$ models a process where all runs are bound to produce a combination of events matching A. Let a and b be labelled by l_1 and l_2 respectively. Sequent (3.i) below is to be read: any l_1 action must be followed by an l_2 action, while sequent (3.ii) forbids l_2 actions to be preceded by (to depend causally on) l_1 actions.



(i)



(ii)

(3)

By similar statements it is possible to describe the behaviour of concurrent programs axiomatically. This is shown in Section 6 where we shall develop further intuition on the meaning of sequents.

4 Configuration Theories

Definition 7 *A* configuration theory *is a set of sequents which is closed under the rule schemes of Table 1.*

The rule [l-weak] only allows the premises of a sequent to be weakened by the empty poset $\emptyset$. Left weakening by an arbitrary poset (as in [r-weak]) would be unsound, as in fact it would allow the inference of $A, A \vdash_{id;id} A$ from $A \vdash_{id} A$ (see Example 6). Rule [r-cut] is a special case of [$\Diamond$], which was introduced as a formation rule in Section 2. Indeed [$\Diamond$] can be derived from [r-cut]. Note that [$\Diamond$] has an obvious dual, which is a general form (and is derivable from) [l-cut].

[true]	$\dfrac{}{\vdash \emptyset}$	[iso]	$\dfrac{}{A \vdash_{\phi} B}$ (ϕ is iso)
[l-weak]	$\dfrac{\Gamma \vdash_{\rho} \Delta}{\Gamma, \emptyset \vdash_{\rho;\emptyset} \Delta}$	[r-weak]	$\dfrac{\Gamma \vdash_{\rho} \Delta}{\Gamma \vdash_{\rho,\sigma} \Delta, A}$ $(*)$
[l-contr]	$\dfrac{\Gamma, A, A \vdash_{\rho;\sigma;\sigma} \Delta}{\Gamma, A \vdash_{\rho;\sigma} \Delta}$	[r-contr]	$\dfrac{\Gamma \vdash_{\rho,\sigma,\sigma} \Delta, A, A}{\Gamma \vdash_{\rho,\sigma} \Delta, A}$
[l-exc]	$\dfrac{\Gamma, A, B, \Pi \vdash_{\rho;\sigma;\tau;\theta} \Delta}{\Gamma, B, A, \Pi \vdash_{\rho;\tau;\sigma;\theta} \Delta}$	[r-exc]	$\dfrac{\Gamma \vdash_{\rho,\sigma,\tau,\theta} \Delta, A, B, \Pi}{\Gamma \vdash_{\rho,\tau,\sigma,\theta} \Delta, B, A, \Pi}$
[l-cut]	$\dfrac{\Pi \vdash_{\tau} A \qquad \Gamma, A \vdash_{\rho;\sigma} \Delta}{\Gamma, \Pi \vdash_{\rho;\tau\sigma} \Delta}$	[r-cut]	$\dfrac{A \vdash_{\tau} \Pi \qquad \Gamma \vdash_{\rho,\sigma} \Delta, A}{\Gamma \vdash_{\rho,\sigma\tau} \Delta, \Pi}$

$(*)$ where σ is a column vector of monos $\sigma_i : \Gamma_i \rightarrowtail A$.

Table 1. Structural Rules

A *model* of a configuration theory is a structure which satisfies all sequents of the theory.

Theorem 8 *The rules of Table 1 are sound.*

Proof. It is required to prove that, if a configuration structure satisfies the premises of a rule, then it also satisfies the conclusion. We just prove the statement for the left and right cut rules. The argument is similar for the others.

[l-cut]. Let $\mathcal{C}$ satisfy the sequents $\Pi \vdash_{\tau} A$ and $\Gamma, A \vdash_{\rho;\sigma} \Delta$, let $C \in \mathcal{C}$ be a configuration, and let $\upsilon : \Gamma \to C$ and $\pi : \Pi \to C$ be matrices of monos. Satisfaction of τ yields an inclusion $C \hookrightarrow D$ and a map $q : A \rightarrowtail D$ making the $(*)$ square of diagram (4) commute for all i. Then, considering q in conjunction with the maps $\Gamma_j \xrightarrow{\upsilon_j} C \hookrightarrow D$, satisfaction of $\rho;\sigma$ yields an inclusion $D \hookrightarrow D'$, a component Δ_k and a map $q' : \Delta_k \rightarrowtail D'$ making all the rest of diagram (4) commute for all i and j. Since $\tau_i \sigma_k = (\tau\sigma)_{ik}$, we conclude that $\mathcal{C}$ satisfies $\Gamma, \Pi \vdash_{\rho;\tau\sigma} \Delta$ as required.

[r-cut]. Let $\mathcal{C}$ satisfy the sequents $A \vdash_{\tau} \Pi$ and $\Gamma \vdash_{\rho,\sigma} \Delta, A$, let $C \in \mathcal{C}$ be a configuration, and let $\upsilon : \Gamma \to C$ be a matrix of monos. Satisfaction of ρ, σ yields an injection $C \hookrightarrow D$ and moreover, for all $\Gamma_i \in \Gamma$, a commuting square as $(\star)$ below:

$$\begin{tikzcd}
\Gamma_i \arrow[r, "\xi_i"] \arrow[d, "\upsilon_i"'] \arrow[dr, phantom, "(\star)"] & X \arrow[d, "q"] & & A \arrow[r, "\tau_k"] \arrow[d, "q"'] \arrow[dr, phantom, "(\star\star)"] & \Pi_k \arrow[d, "q'"] \\
C \arrow[r, hook] & D & & D \arrow[r, hook] & D'
\end{tikzcd}$$

where either $X = A$ and $\xi_i = \sigma_i$, or $X = \Delta_j$ for a component $\Delta_j \in \Delta$, and $\xi_i = \rho_{ij}$. In the last case the result follows immediately. Otherwise $X = A$ and, since τ is satisfied, there exists a component $\Pi_k \in \Pi$, an inclusion $D \hookrightarrow D'$ and a map $q' : \Pi_k \rightarrowtail D'$ such that the diagram $(\star\star)$ above commutes. Pasting $(\star)$ and $(\star\star)$ we get the required instance of diagram (1), where $\sigma_i\tau_\kappa = (\sigma\,\tau)_{ik}$.

□

$$\begin{tikzcd}
\Gamma_j \arrow[drrr, "\rho_{jk}"] \arrow[ddr, "\upsilon_j"'] & & & \\
& \Pi_i \arrow[r, "\tau_i"] \arrow[d, "\pi_i"] \arrow[dr, phantom, "(*)"] & A \arrow[r, "\sigma_k"] \arrow[d, "q"] & \Delta_k \arrow[d, "q'"] \\
& C \arrow[r, hook] & D \arrow[r, hook] & D'
\end{tikzcd} \tag{4}$$

5 Completeness

There are valid sequents which cannot be derived from the inference rules of Table 1. One is Diagram (2) of Section 3. In this section we obtain a complete calculus at the cost of constraining sequents (but not the models) to be *finite*, that is, made of finite posets. Indeed, since Diagram (2) is finite, new rules are needed to achieve completeness. Note that the Java axioms of Section 6 are finite.

First we introduce a notion of order (in fact a preorder) on matrices of poset maps which is somewhat analogous to the notion of *rank* in linear algebra. Let $\rho : \Gamma \to A_1, \dots A_m$ and $\sigma : \Gamma \to B_1, \dots B_n$ be matrices of posets. We write $\rho \leq_\Gamma \sigma$ (omitting Γ when understood) if there exist a function on indices $f : \{1, \dots n\} \to \{1, \dots m\}$ and a family $\{\phi_j\}$ of monos $\phi_j : A_{f(j)} \rightarrowtail B_j$, $j = 1 \dots n$, such that $\sigma_{ij} = \rho_{if(j)}\phi_j$, for all i. In this case we say that f and $\{\phi_j\}$ *witness* $\rho \leq \sigma$. The relation $\leq$ is reflexive and transitive. We call *equivalent* two matrices ρ and σ such that $\rho \leq \sigma$ and $\sigma \leq \rho$. The equivalence class of ρ is written $[\rho]$.

Proposition 9 *Let $\Gamma \vdash_\rho \Delta$ and $\Gamma \vdash_\sigma \Pi$ be sequents: $\rho \leq \sigma$ holds if and only if, whenever a structure satisfies σ, it also satisfies ρ.*

Proof. *If*: Each $\Pi_i \in \Pi$ (viewed as a configuration structure) satisfies σ, and so it must satisfy ρ. Hence, considering the interpretation $\sigma_{(\_)i} : \Gamma \to \Pi_i$, there exist a $\Delta_{f(i)} \in \Delta$ and a mono $\phi_i : \Delta_{f(i)} \rightarrowtail \Pi_i$ such that $\sigma_i = \rho_{f(i)}\phi_i$ as required.

Only if: Let $\rho \leq \sigma$, let $\mathcal{C}$) satisfy σ and let $\pi : \mathit{\Gamma} \to C \in \mathcal{C}$. There must exist a $D \in \mathcal{C}$, an inclusion $u : C \hookrightarrow D$, a $\mathit{\Pi}_k \in \mathit{\Pi}$ and a mono $q : \mathit{\Pi}_k \rightarrowtail D$ such that $\pi_i u = \sigma_{ik} q$, for all i. Since $\rho \leq \sigma$, there must exist $\mathit{\Delta}_{f(k)} \in \mathit{\Delta}$ and a mono $\phi_k : \mathit{\Delta}_{f(k)} \rightarrowtail \mathit{\Pi}_k$ such that $\rho_{if(k)}\phi_k = \sigma_{ik}$ for all i; hence $\pi_i u = \sigma_{ik} q = \rho_{if(k)}\phi_k q$ as required. □

By the above result the inference rule [sub] below is sound. This rule is used in Section 6. Moreover, since any matrix $\sigma : \mathit{\Gamma} \to \mathit{\Pi}$ is such that $\sigma \leq \epsilon$, where $\epsilon : \mathit{\Gamma} \to \varepsilon$ is the empty matrix and ε is the empty sequence, the *falsum* rule below is a special case of [sub]. Similarly, [sub] subsumes [iso], [r-weak], [r-contr] and [r-exc].

$$\text{[sub]} \quad \frac{\mathit{\Gamma} \vdash_\rho \mathit{\Delta}}{\mathit{\Gamma} \vdash_\sigma \mathit{\Pi}} \quad (\sigma \leq \rho) \qquad\qquad \text{[falsum]} \quad \frac{\mathit{\Gamma} \vdash}{\mathit{\Gamma} \vdash_\sigma \mathit{\Pi}}$$

Definition 10 *A matrix μ of size $m \times n$ is called* minimal *in its equivalence class when, for all $\rho \in [\mu]$ of size $m \times n'$, $n \leq n'$.*

We show that, when considering matrices $\mathit{\Gamma} \to \mathit{\Delta}$ where all components of $\mathit{\Gamma}$ and $\mathit{\Delta}$ are *finite*, all minimal matrices in an equivalence class are *isomorphic.* In the rest of this section we assume, unless otherwise stated, that matrices are made of maps between finite posets. This assumption should also reassure the concerned reader that the side condition of [sub] can be checked effectively.

Proposition 11 *Let $\rho : \mathit{\Gamma} \to \mathit{\Delta}$ and $\mu : \mathit{\Gamma} \to \mathit{\Pi}$ be equivalent matrices, with μ of size $m \times n$ minimal in $[\rho]$. Then $\rho \leq \mu$ is witnessed by a family of* isomorphisms *$\phi_j : \mathit{\Delta}_{f(j)} \rightarrowtail \mathit{\Pi}_j$.*

Proof. Let ρ have size $m \times n'$ and let $\rho \leq \mu$ by $f : \{1, \ldots n\} \to \{1, \ldots n'\}$ and by a family of monos $\phi_j : \mathit{\Delta}_{f(j)} \rightarrowtail \mathit{\Pi}_j$. Let ρ' be the matrix obtained from ρ by deleting all the columns $\rho_{(\_)k} : \mathit{\Gamma} \to \mathit{\Delta}_k$ such that $k \neq f(j)$ for all j. Clearly $\rho \leq \rho'$, and moreover $\rho' \leq \mu$ by the same f and ϕ_j. Hence $\rho' \in [\mu]$. It follows that f must be injective, otherwise its image would be smaller than n, thus contraddicting the minimality of μ. So f is a bijection. Let $\mu \leq \rho'$ by a function g and a family $\psi_i : \mathit{\Pi}_{g(i)} \rightarrowtail \mathit{\Delta}_i$. By the same argument as above g must be a bijection, and hence all nodes in the bipartite directed graph whose edges are the ϕ_j and the ψ_i belong to (exactly) one cycle, which implies, from the remark at the the beginning of Section 2, that all ϕ_j are isos. □

A consequence of this result is that if two $m \times n$ matrices $\mathit{\Gamma} \to \mathit{\Delta}$ and $\mathit{\Gamma} \to \mathit{\Pi}$ are equivalent and minimal, there exists a family of n isomorphisms $\mathit{\Delta}_i \xrightarrow{\cong} \mathit{\Pi}_i$ through which all their components factorise. Hence, by a slight mathematical abuse, we say *the* minimal matrix of an equivalence class.

We can now define an operation that yields all possible mergings of two finite posets B and C which are consistent on some common intersection A.

Lemma 12 *Let A, B, C be finite posets and let $p : A \rightarrowtail B$ and $q : A \rightarrowtail C$ be monos. There exists a (possibly empty) matrix $(\pi;\tau) : B, C \to \Pi$ such that $p\,\pi_i = q\,\tau_i$ for all i, and moreover $(\pi;\tau) \leq (r;s)$ for all $r : B \rightarrowtail D$ and $s : C \rightarrowtail D$ such that $p\,r = q\,s$.*

Sketch of proof. Let K be any set of cardinality $k = |B|+|C|$ and let $K_1, \ldots K_n$ be the set of all posets whose underlying set is K. Their number (n) is $(k-1)3k/2$. For $j = 1 \ldots n$, consider the (finite) set of diagrams $B \rightarrowtail K_j \leftarrowtail C$ which commute with $(p;q)$. The components of Π are the images of all such diagrams, while π and τ are made of the injections. $\square$

Of course, all matrices with the above property are equivalent. We let $\mu(p,q)$ be the minimal one of the equivalence class. By using this construction we can now introduce a new rule of inference which yields the *extension* of a sequent $\Gamma, A \vdash \Delta$ along a mono $A \rightarrowtail C$:

$$\text{[extend]} \quad \frac{\Gamma, A \vdash_{\rho;\sigma} B_1, \ldots B_n}{\Gamma, C \vdash_{(\rho \,\Diamond\, \pi);\tau} \Pi^{(1)}, \ldots \Pi^{(n)}} \quad (*)$$

($*$) where $\pi = [\pi^{(1)}, \ldots \pi^{(n)}]$, $\tau = [\tau^{(1)}, \ldots \tau^{(n)}]$, $q : A \rightarrowtail C$ and, for all i, $(\pi^{(i)}, \tau^{(i)}) = \mu(\sigma_i, q) : (B_i, C) \to \Pi^{(i)}$.

Note that [extend] only makes sense in a calculus of finite posets, where $\mu(\_,\_)$ is defined.

Proposition 13 [extend] *is sound.*

Proof. Let $\Gamma, A \vdash_{\rho;\sigma} B_1, \ldots B_n$ be satisfied by $\mathcal{C}$), let $q : A \rightarrowtail C$ be a mono and let $(\zeta;p) : (\Gamma, C) \to D$ be an interpretation of (Γ, C) in a configuration $D \in \mathcal{C}$. Since $(\zeta; q\,p) : (\Gamma, A) \to D$, there exist $D' \in \mathcal{C}$, an inclusion $u : D \hookrightarrow D'$, a B_k and a mono $r : B_k \rightarrowtail D'$ such that $\sigma_k r = q\,p\,u$ and, for all i, $\rho_{ik} r = \zeta_i u$. Let $\mu(\sigma_k, q) = (\pi^{(k)}; \tau^{(k)}) : (B_k, C) \to \Pi^{(k)}$. Since $(\pi^{(k)}; \tau^{(k)}) \leq (r; p\,u)$, there exists $\Pi^{(k)}_h \in \Pi^{(k)}$ and a mono $\phi : \Pi^{(k)}_h \rightarrowtail D'$ such that $\pi^{(k)}_h \phi = r$ and $\tau^{(k)}_h \phi = p\,u$. Moreover, let $\rho \,\Diamond\, \pi = (\rho_{(\_)1}\pi^{(1)}, \ldots \rho_{(\_)n}\pi^{(n)})$. The h-component of $(\rho \,\Diamond\, \pi)^{(k)}$ is $\rho_{(\_)k}\,\pi^{(k)}_h : \Gamma \to \Pi^{(k)}_h$, and hence $\rho_{ik}\,\pi^{(k)}_h \phi = \rho_{ik}\,r = \zeta_i\,u$ as required.

$\square$

Lemma 14 [extend] *preserves minimality.*

Proof. With no loss of generality we develop the argument for $A, B \vdash_\rho C, D$. Let $A, F \vdash_\sigma \Pi$ be the extension of ρ along a mono $q : B \rightarrowtail F$, and suppose that σ is not minimal. Let $\nu \in [\sigma]$ be minimal and let f be the funcion on indices witnessing $\sigma \leq \nu$. There must be a $\Pi_k \in \Pi^{(i)}$ such that $k \neq f(j)$ for all j. The matrix σ' obtained from σ by deleting $\sigma_{(\_)k}$ must still be in $[\sigma]$. Hence there must exist $\Pi_h \in \Pi^{(l)}$ and $p : \Pi_h \rightarrowtail \Pi_k$ through which the interpretation $\sigma_{(\_)k}$ of (A, F) factorises. However, it must be $l \neq i$, because otherwise $\mu(\rho_{2i}, q)$ would not be minimal. But then $\rho_{(\_)i} \simeq \rho_{(\_)l}$ contraddicting the minimality of ρ.

$\square$

Theorem 15 (completeness) *The system of finite sequents which includes the axioms of Table 1 and* [extend] *is complete.*

Proof. Let $A_1, \dots A_m \vdash_\rho \Delta$ be a valid sequent. It is required to prove that ρ is derivable. Consider the derivation:

$$
\begin{array}{rcl}
\text{[l-weak]} & \dfrac{A_1 \vdash_{id} A_1}{A_1, \emptyset \vdash_{id;\emptyset} A_1} & \\
\text{[extend]} & \dfrac{A_1, \emptyset \vdash_{id;\emptyset} A_1}{A_1, A_2 \vdash_\pi \Pi} & (\emptyset \rightarrowtail A_2) \\
\text{[l-weak]} & \dfrac{A_1, A_2 \vdash_\pi \Pi}{A_1, A_2, \emptyset \vdash_{\pi;\emptyset} \Pi} & \\
 & \vdots & \\
\text{[extend]} & \dfrac{\qquad\qquad}{A_1, \dots A_m \vdash_\sigma B_1, \dots B_n} & (\emptyset \rightarrowtail A_m)
\end{array}
$$

where the A_i are introduced one-by-one by m applications of [l-weak] and [extend]. Since these rules are sound and $A_1 \vdash_{id} A_1$ is valid, then so is also σ. Hence, by Proposition 9, $\sigma \in [\rho]$. Moreover [l-weak] preserves minimality trivially, while [extend] does so by Proposition 14. Since $A_1 \vdash_{id} A_1$ is minimal, then so is also σ. Therefore, from Proposition 11, $\rho \leq \sigma$ is witnessed by a family of isomorphisms $\phi_j : \Delta_{f(j)} \xrightarrow{\simeq} B_j$ such that $\rho_{if(j)}\, \phi_j = \sigma_{ij}$. Then, by n applications of [iso] and [r-cut] we derive:

$$
\text{[r-cut]}^n \; \frac{A_1, \dots A_m \vdash_\sigma B_1, \dots B_n \qquad B_i \vdash_{\phi_i^{-1}} \Delta_{f(i)}}{A_1, \dots A_m \vdash_{\sigma_{(\_)f(1)}, \dots \sigma_{(\_)f(n)}} \Delta_{f(1)}, \dots \Delta_{f(n)}}
$$

The rest of σ can then be adjoined by [r-weak]. □

6 The Theory of Java

The interaction of threads (lightweight processes) and memory in Java is described in the language specification [GJS96, Ch. 17] by means of eight kinds of actions. Besides *Lock* and *Unlock*, which we do not consider here, they are: *Use*, *Assign*, *Load*, *Store*, *Read* and *Write*. These are abstractions over corresponding Java bytecode instructions. We let u, a, l, s, r and w stand for events labelled respectively by these types of actions. Each thread has a *working memory* where private copies of shared variables are cached. Threads operate on their own working memory by *Use* and *Assign* actions. For example, a thread θ performing an assignment $x = x + 1$, first *uses* the content of its working copy of x to compute $x + 1$, and then *assigns* the computed value v to x. However, v is not available to other threads unless θ decides (nondeterministically) to *store* the current value of its copy x to the main memory, where the *master copies* of all variables reside. The *Store* action is just a message sent *asynchronously* by θ to the main memory: the actual writing of v in the master copy of x is performed by the main memory (possibly at a later time) with a *Write* action. Similarly *Read* and *Load* are used for a loosely coupled copying of data from the main memory to a thread's working memory.

Following [CKRW98] we label events by 4-tuples of the form (α, θ, x, v), where $\alpha \in \{\mathit{Use}, \mathit{Assign}, \mathit{Load}, \mathit{Store}, \mathit{Read}, \mathit{Write}\}$, θ is a thread identifier, x is a variable and v is a value. We write $e : l$ to mean that event e has label l. Label components are omitted when undestood or irrelevant. Hence, if $u_{x1} : (\mathit{Use}, \zeta, x, 1)$, then u_{x1} represents the use of variable x, whose current value is 1, by a thread ζ, as in the example below for evaluating the right hand side of the assignment $y = x$, while $a : (\mathit{Assign}, y)$ stands for an assignment of an unspecified value to y by an unspecified thread.

Table 2 shows a possible order of events which may occur when two threads θ and ζ, running in parallel left to right, execute respectively $(x = 1; x = y;)$ and $(y = 2; y = x;)$. The events are labeled as follows: $a_{x1} : (\mathit{Assign}, \theta, x, 1)$, $s_{x1} : (\mathit{Store}, \theta, x, 1)$, $l_{y2} : (\mathit{Load}, \theta, y, 2)$, $u_{y2} : (\mathit{Use}, \theta, y, 2)$, $a_{x2} : (\mathit{Assign}, \theta, x, 2)$, $w_{x1} : (\mathit{Write}, \theta, x, 1)$, $r_{x1} : (\mathit{Read}, \zeta, x, 1)$, $w_{y2} : (\mathit{Write}, \zeta, y, 2)$, $r_{y2} : (\mathit{Read}, \theta, y, 2)$, $a_{y2} : (\mathit{Assign}, \zeta, y, 2)$, $s_{y2} : (\mathit{Store}, \zeta, y, 2)$, $l_{x1} : (\mathit{Load}, \zeta, x, 1)$, $u_{x1} : (\mathit{Use}, \zeta, x, 1)$, $a_{y1} : (\mathit{Assign}, \zeta, y, 1)$. The execution ends with $x = y = 2$ in the working memory of θ and $x = y = 1$ in the working memory of ζ. Note that there is no causal dependency between actions performed by the memory on different variables, as between w_{x1} and w_{y2}. The ordering is legal according to the informal specification given in [GJS96].

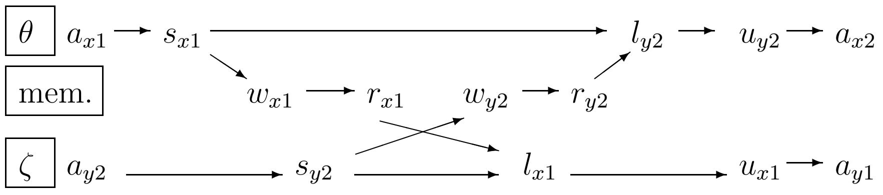
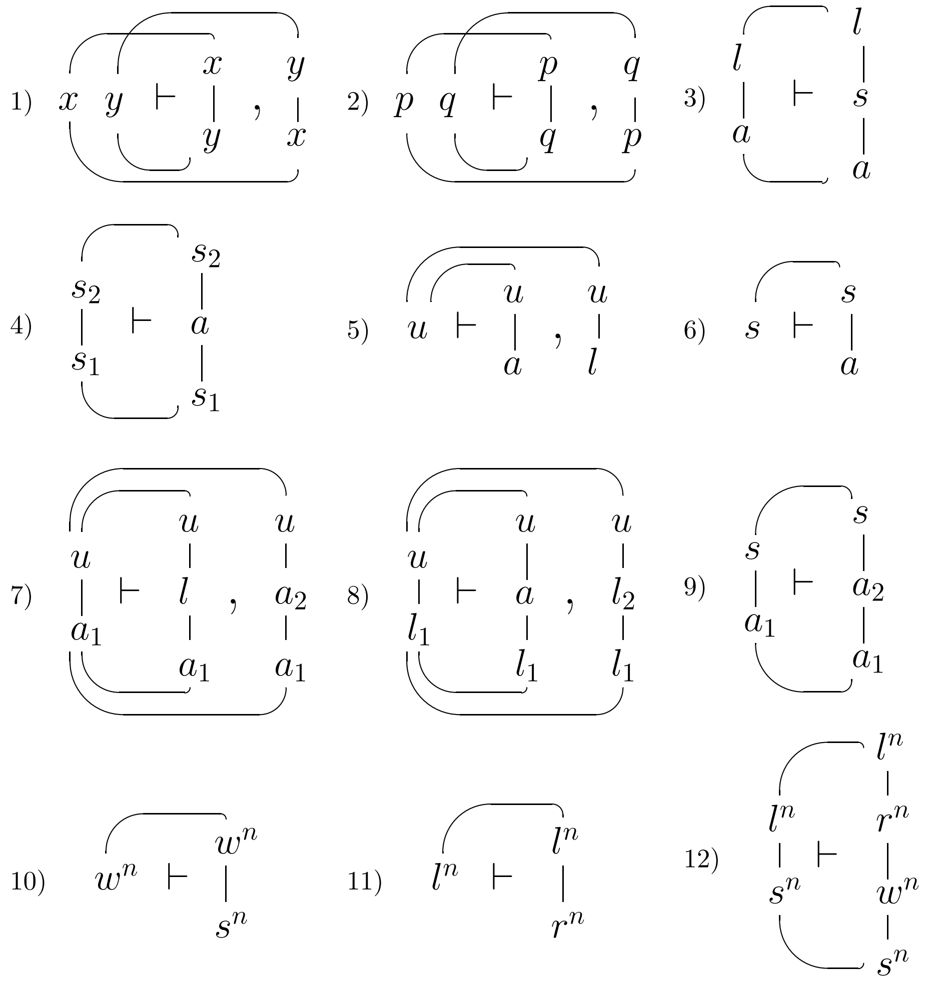

$$\begin{tikzcd}
\boxed{\theta} & a_{x1} \arrow[r] & s_{x1} \arrow[rrrr] \arrow[dr] & & & & l_{y2} \arrow[r] & u_{y2} \arrow[r] & a_{x2} \\
\boxed{\text{mem.}} & & & w_{x1} \arrow[r] & r_{x1} \arrow[drr] & w_{y2} \arrow[r] & r_{y2} \arrow[u] & & \\
\boxed{\zeta} & a_{y2} \arrow[rrr] & & & s_{y2} \arrow[rr] \arrow[ur] & & l_{x1} \arrow[r] & u_{x1} \arrow[r] & a_{y1}
\end{tikzcd}$$

Table 2. $(x = 1; x = y;)\ ||\ (y = 2; y = x;)$

Below we list 12 formal axiom schemes describing the protocol of memory and thread interaction in Java. They are subject to the side conditions given below, specifying what labels are to be attached to each event. By w^n we mean a *totally* ordered set $\{w_1 \leq w_2 \leq \dots w_n\}$ of n events of type *Write*. Similarly for s^n, l^n and r^n. We do not consider synchronization by *lock* and *unlock*: the full theory, including synchronization by *lock* and *unlock* (6 additional axioms), is available at `http://cenciarelli.dsi.uniroma1.it/~cencia`.

As an example we explain axiom scheme (3). It represents all sequents of that form where $a : (\mathit{Assign}, \theta, x, v)$, $s : (\mathit{Store}, \theta, x, v)$ and $l : (\mathit{Load}, \theta, x)$ (the value being loaded in x is irrelevant). Hence: a *Store* action by θ on a variable x must intervene between an *Assign* by θ of x and a subsequent *Load* by θ of x. This is because a "thread is not permitted to lose its most recent assign" [GJS96, § 17.3].



The above sequents express the following requirements:

(1) $x : (\alpha, \theta)$ and $y : (\alpha', \theta)$, $\alpha, \alpha' \in \{\mathit{Use}, \mathit{Assign}, \mathit{Load}, \mathit{Store}\}$ (these are called *thread actions*). Intuitively, this axiom means that *the actions performed by any one thread are totally ordered* [GJS96, §17.2].

(2) $p : (\beta, \theta, x)$ and $q : (\beta', \theta, x)$, where $\beta, \beta' \in \{\mathit{Read}, \mathit{Write}\}$ (*memory actions*). *The actions performed by the main memory for any one variable are totally ordered* [ibid. §17.5].

(4) $s_1 : (\mathit{Store}, \theta, x, v_1)$, $s_2 : (\mathit{Store}, \theta, x, v_2)$ and $a : (\mathit{Assign}, \theta, x, v_2)$. *A thread is not permitted to write data from its working memory back to main memory for no reason* [ibid. §17.3].

(5) and (6) $u : (\mathit{Use}, \theta, x, v)$, $a : (\mathit{Assign}, \theta, x, v)$, $l : (\mathit{Load}, \theta, x, v)$ and $s : (\mathit{Store}, \theta, x, v)$. *Threads start with an empty working memory and new variables* *are created only in main memory and are not initially in any thread's working memory* [ibid. §17.3].

(7) and (8) $a_1 : (\mathit{Assign}, \theta, x, v_1)$, $u : (\mathit{Use}, \theta, x, v_2)$, with $v_2 \neq v_1$, l and $l_2 : (\mathit{Load}, \theta, x, v_2)$, a_2 and $a : (\mathit{Assign}, \theta, x, v_2)$, $l_1 : (\mathit{Load}, \theta, x, v_1)$. *A Use action transfers the contents of the thread's working copy of a variable to the thread's execution engine* [ibid. §17.1].

(9) $a_1 : (\mathit{Assign}, \theta, x, v_1)$, $s : (\mathit{Store}, \theta, x, v_2)$, $a_2 : (\mathit{Assign}, \theta, x, v_2)$ and $v_2 \neq v_1$. *A Store action transmits the contents of the thread's working copy of a variable to main memory* [ibid. §17.1].

(10) and (11) $w_i : (\mathit{Write}, \theta, x, v_i)$, $s_i : (\mathit{Store}, \theta, x, v_i)$, $l_i : (\mathit{Load}, \theta, x, v_i)$ and $r_i : (\mathit{Read}, \theta, x, v_i)$, for $i = 1 \ldots n$. *Each Load or Write action is uniquely paired with a preceding Read or Store action respectively. Matching actions bear identical values* [ibid. §17.2,§17.3].

*(***12)** Labels as above. *The actions on the master copy of any given variable on behalf of a thread are performed by the main memory in exactly the order that the thread requested* [ibid. §17.3].

The Java language specification states that *a thread is not permitted to write data from its working memory back to main memory for no reason.* Axiom (4) alone does not seem to guarantee this property, and in fact [GJS96, §17.3] introduces explicitly a similar clause requiring that an assignment exists in between a *load* and a subsequent *store*. This is expressed formally by a version of axiom (4), call it (4-bis), where s_1 is replaced by $l : (\mathit{Load}, \theta, x, v_1)$. In [CKRW98] we proved that (4-bis) follows from the other axioms (in a non-obvious way). Here we are able to derive this sequent formally in the theory of Java. However, to do so we need a new rule, [grd], stating that configurations are *grounded*, that is: there are no infinite descending chains of events (see Proposition 3).

Let $s : A \rightarrowtail B$ and let $t : A \rightarrowtail D$. We write $t \ll s$ if there exists $r : A \rightarrowtail B$, $r \neq s$, such that for all $a \in A$:

- if $r(a) < s(a)$ then $D \uparrow t(a) \subseteq t(A)$;
- if $r(a) > s(a)$ then $D \downarrow t(a) \subseteq t(A)$ and there exists $a' \in A$ such that $r(a) < r(a') \leq s(a')$.

When A, B and D are *chains*, that is totally ordered sets, the condition expressed by $\ll$ allows the following inference:

$$[\text{grd}] \quad \frac{A \vdash_{\rho,\sigma} D, B}{A \vdash_{\rho} D} \quad (\text{if } A,\ B,\ D \text{ are chains and } \rho \ll \sigma)$$

By suitably generalising the relation $\ll$, this rule can be extended to arbitrary posets. The proof of soundness for [grd] is rather lengthy. Here we just give the intuition with an example: Let $A = \{a\}$, $B = \{a_1 < a_2\}$, $D = \{b < a_3\}$, let a and the a_i be labelled by l and b by $l' \neq l$. Moreover, let $\sigma(a) = a_2$ and $\rho(a) = a_3$.

It is easy to check that $\rho \ll \sigma$, where the required $r : A \rightarrowtail B$ is $r(a) = a_1$. To wit, σ can be viewed as iteratively "generating" events (namely a_2) below a. But iteration cannot go on indefinitely: the reader can verify that any configuration C of a structure $\mathcal{C}$ satisfying (ρ, σ) must feature a chain of l-actions preceeded by an $l\,'$-action (postulated by ρ; this justifies the notation $\rho \ll \sigma$), or otherwise no l-actions at all. Then, any interpretation $A \to C$ factorising through σ will also factorise through ρ, which means that $\mathcal{C}$ satisfies ρ.

Derivation of (4-bis). Events are meant as labelled according to the convention introduced above. Let $A = \{a < l < s\}$, $B = \{a_1 < s_1 < a_2 < l_1 < s_2\}$ and $D = \{a_3 < s_3 < l_2 < a_4 < s_4\}$. Let $\sigma : A \rightarrowtail B$ be the map $\sigma(a) = a_1$; the rest of σ is forced by the labels, and so is $\rho : A \rightarrowtail D$. The sequent $A \vdash_{\rho,\sigma} D, B$ can be derived from axioms (1), (3) and (4) using [extend] and [r-cut]. Since $\rho \ll \sigma$, [grd] yields $A \vdash_\rho D$. Moreover, let $E = \{l_3 < s_5\}$ and $F = \{l_4 < a_5 < s_6\}$, and let $\tau : E \rightarrowtail A$ and $\pi : E \rightarrowtail F$ be the obvious maps. From (1) and (6) we derive $E \vdash_{\tau,\pi} A, F$ and hence, by [r-cut], $E \vdash_{\tau\rho,\pi} D, F$. Since $\pi \leq (\tau\rho, \pi)$, [sub] yields $E \vdash_\pi F$ as required. □

Acknowledgements

Thanks to Alexander Knapp and Anna Labella for the many useful discussions.

References

[Cen00] P. Cenciarelli. Event Structures for Java. In S. Drossopoulou, S. Eisenbach, B. Jacobs, G. Leavens, P. Mueller, and A. Poetzsch-Heffter, editors, *Proceedings of the ECOOP 2000 Workshop on Formal Techniques for Java Programs, Cannes, France*, June 2000.

[CKRW98] P. Cenciarelli, A. Knapp, B. Reus, and M. Wirsing. An Event-Based Structural Operational Semantics of Multi-Threaded Java. In J. Alves-Foss, editor, *Formal Syntax and Semantics of Java*, 1523 LNCS. Springer, 1998.

[GG90] R.J. van Glabbeek and U. Goltz. Refinement of Actions in Causality Based Models. In W.P. de Roever J.W. de Bakker and G. Rozenberg, editors, *LNCS 430*, pages 267–300. Springer-Verlag, 1990.

[GG01] R.J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37:229–327, 2001.

[GJS96] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification.* Addison-Wesley, 1996.

[GP95] R.J. van Glabbeek and G.D. Plotkin. Configuration structures (extended abstract). In D. Kozen, editor, *Proceedings of LICS'95*, pages 199–209. IEEE Computer Society Press, June 1995.

[NPW81] M. Nielsen, G.D. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains: Part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

[Win82] G. Winskel. Event Structure Semantics of CCS and Related Languages. *Springer LNCS*, 140, 1982. Proceedings ICALP'82.

[Win87] Glynn Winskel. Event Structures. In G. Rozenberg W. Brauer, W. Reisig, editor, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, number 255 in LNCS. Springer-Verlag, 1987.