Models for Open Transactions*

Roberto Bruni and Ugo Montanari

Computer Science Department, University of Pisa, Italy

Abstract. Loosely coupled interactions permeate modern distributed systems, where autonomous applications need to collaborate by dynamical assembly. We can single out three different phases occurring in every collaboration: 1) negotiation of some sort of contracts, mediating the needs of prospective participants; 2) acceptance or rejection of the contract; 3) contract-guarantee execution. The above scheme, called NCE for short (Negotiation, Commit, Execution), covers a wide range of situations, ranging from sessions and transactions to proof-carrying code. In the paper we concentrate on the notion of open transaction and on Zero-Safe Nets, a model developed by the authors for modelling long transactions. We extend the latter to cover the three-phase process above.

1 Introduction

Recent years have witnessed a progressive shifting of importance from the traditional concept of safe, overly controlled computation to open-ended, loosely coupled interactions, which permeate modern distributed systems. In particular, features such as concurrency, dynamicity, and adaptability inevitably arise in global computing frameworks based on wide-area networks, where separately developed, autonomous applications need to collaborate by dynamical assembly.

An emblematic example is the service-oriented computing (SOC) paradigm, where service abstractions are published in public repositories, which can be queried by other services for discovering convenient partners to interact with. Since it is desirable that interactions are carried out in a safe way for all participants, different notions of contracts have emerged to assign responsibilities to each participant in a non-ambiguous way and to give suitable run-time guarantees whenever all participants respect the contract. Formally, this involves answers to two main question: 1) Are the service abstractions compatible with the contract? 2) Is each participant consistent with its abstract description? Note that here we are often concerned with the operational behaviour of services, not just with functional aspects. Moreover, the notions of compatibility and consistency should be such that whenever 1) and 2) are answered in the affirmative, then the overall interaction is guaranteed to be sound (e.g. absence of deadlocks, or type-safe communications, or absence of livelock under fair scheduling assumptions). While traditionally all checks are performed statically, the challenge posed by SOC is moving them to run-time with dynamic assembly.

* Research supported by the EU within the FET-GC II Integrated Project IST-2005-016004 Sensoria and by the Italian FIRB Project Tocai.it.

We can single out three different phases occurring in every collaboration: 1) negotiation; 2) commit; 3) execution. In (1) the prospective participants negotiate some guarantees in order to define a sort of contract. In (2) each participant can either accept or reject the contract. If they accept, the contract will bind their behaviours in (3) to guarantee a globally correct execution. The scheme given by the phases (1-3) is called NCE and it covers a wide range of situations like transaction processing (phases 1-2), session-based interactions (phases 2-3), and applications of proof-carrying code (phases 1 and 3). Necessarily we need to allow part of the verification to be done also at run-time, on the basis of both statically and dynamically negotiable information.

In this paper we deal with multi-party interactions, where the participants can negotiate the interaction protocol to follow according to their abstract behaviour. The outcome of the negotiation is a (possibly non-deterministic and concurrent) well-behaving, global contract that binds the admissible interactions, thus fixing exact responsibilities in case of faulted or misbehaving execution.

Our formalisation relies on Petri nets and it builds on classical results from concurrency theory and workflow management. In particular, it takes inspiration from Zero-Safe nets [1], unfolding construction [2, 3], and workflow nets [4]. Roughly, the procedure can be outlined as follows: each participant describes its admissible behaviour as a Zero-Safe net; the composition of all Zero-Safe nets is unfolded as a non-deterministic process; the unfolding is suitably pruned (e.g., by removing faulted situations) and presented as a contract K; if accepted by all participants, then their stable tokens are tagged according to K and the firing of their transitions apply local consistency checks on such tags. The main result establishes that any such execution cannot deadlock.

Synopsis. In § 2 we fix the notation and the key concepts needed in the rest of the paper. In § 3 we outline the various formal steps of our approach, grouping them according to the three-phases NCE classification. We deal with multi-party interactions, but due to space limitation, only two-parties (buyer and seller) are considered in our running example. Some concluding remarks are in § 4.

2 Preliminaries

Petri nets. Place/Transition Petri nets (PT nets) are bipartite graphs that represent some kind of concurrent automata. Formally, a PT *net* is a triple $N = (S, T, F)$ where S is the set of places, T is the set of transitions and $F : (S \times T) \cup (T \times S) \to \mathbb{N}$ is the flow relation. The states of a PT net, called *markings*, are multisets $u : S \to \mathbb{N}$ of places, representing the number of *tokens* in each place. We write a marking as the formal sum $u = \bigoplus_i n_i a_i$, i.e. as an element of the free commutative monoid $S^{\oplus}$ (monoidal composition is defined by $(\bigoplus_i n_i a_i) \oplus (\bigoplus_i m_i a_i) = \bigoplus_i (n_i + m_i) a_i$, with 0 as the unit). Multiset inclusion is written $u \subseteq v$ (if $u(a) \leq v(a)$ for all places a), and multiset difference $v \ominus u$ (it is defined only when $u \subseteq v$, with $(v \ominus u)(a) = v(a) - u(a)$ for all places a).

For any transition t, its *pre-* and *post-set*, written $\mathrm{pre}(t)$ and $\mathrm{post}(t)$ respectively, are the multisets over S such that $\mathrm{pre}(t)(a) = F(a, t)$ and $\mathrm{post}(t)(a) =$

$F(t, a)$, for all $a \in S$. We write $t : u \to v$ for a transition t with $\mathrm{pre}(t) = u$ and $\mathrm{post}(t) = v$ and say that t is *enabled* in the marking w if $\mathrm{pre}(t) \subseteq w$. If $t : u \to v$ is enabled in w, then t can be *fired* in w leading to $w' = w \ominus u \oplus v$ and its firing is written $w[t\rangle w'$. A *firing sequence* from w to w' is a sequence of firings $w_1[t_1\rangle w'_1$, ..., $w_n[t_n\rangle w'_n$ such that $w = w_1$, $w' = w'_n$ and $w_{i+1} = w'_i$ for all $i \in [1, n-1]$. A marking v is *reachable* from u if there is a firing sequence from u to v.

Unfolding. Starting from a net N and a marking u, the *reachability graph* $\mathcal{R}(N, u)$ is a graph whose nodes are all markings reachable from u in N and whose arcs are all triples $(u, t, u \ominus \mathrm{pre}(t) \oplus \mathrm{post}(t))$ such that $\mathrm{pre}(t) \subseteq u$. While the reachability graphs account for the interleaving description of the computational space of N, the interplay between non-determinism, causality and concurrency available in N is accounted for by the so-called *unfolding construction* $\mathcal{U}(N, u)$. Formally, $\mathcal{U}(N, u)$ is a non-deterministic occurrence net (i.e., an acyclic net, where transition pre- and post-markings are sets instead of multisets and where each place has at most one entering arc, i.e., backward conflicts are not allowed), together with a net homomorphism from $\mathcal{U}(N, u)$ to N that tells which places and transitions of the unfolding are instances of the same element of N. Roughly, the transitions of $\mathcal{U}(N, u)$, called *events*, represent all the possible firings of transitions in N in all possible runs of the net, and the places of $\mathcal{U}(N, u)$ are all the possible tokens that can be generated.

For occurrence nets, the notion of *causally dependent*, of *conflicting* and of *concurrent* elements can be represented by the binary relations $\preceq$, $\#$ and $\mathbf{co}(\_, \_)$, respectively. Formally, $\preceq$ is the transitive and reflexive closure of the *immediate precedence* relation $\prec_0 \stackrel{\mathrm{def}}{=} \{(a, t) \mid a \in \mathrm{pre}(t)\} \cup \{(t, a) \mid a \in \mathrm{post}(t)\}$. Letting $t_1 \#_0 t_2 \stackrel{\mathrm{def}}{\Leftrightarrow} t_1 \neq t_2 \ \wedge\ \mathrm{pre}(t_1) \cap \mathrm{pre}(t_2) \neq \emptyset$, binary conflict $\#$ is defined as the minimal symmetric relation that contains $\#_0$ and that is hereditary with respect to $\preceq$ (i.e., such that if $x_1, x_2, y \in S \cup T$ and $x_1 \# x_2$ and $x_1 \preceq y$ then $y \# x_2$). The concurrency relation is defined by $\mathbf{co}(x_1, x_2) \stackrel{\mathrm{def}}{\Leftrightarrow} \neg(x_1 \prec x_2 \vee x_2 \prec x_1 \vee x_1 \# x_2)$ and it is extended to sets of elements by letting $\mathbf{co}(X) \stackrel{\mathrm{def}}{\Leftrightarrow} \forall x_1, x_2 \in X\ \mathbf{co}(x_1, x_2)$.

The net $\mathcal{U}(N, u)$ is defined (up to iso) as the net generated by the rules in Table 1, whose places have the form $\langle a, H, k\rangle$ and whose transitions have the form $\langle t, H\rangle$, where $a \in S$, $t \in T$, H is a set of causes that encodes the history of the element, and k is a positive natural number used to distinguish different tokens with the same history. The top rule introduces a distinguished place (with empty history) for each of the tokens in the initial marking u. The bottom rule adds an event e that represents the firing of a transition that consumes the tokens in Θ. The condition $\mathbf{co}(\Theta)$ rules out all inapplicable firings. The event e introduces the elements in Υ, representing the tokens produced by the corresponding firing.

The elements in $\mathcal{U}(N, u)$ can be stratified according to the depth function $\delta : S_{\mathcal{U}(N,u)} \cup T_{\mathcal{U}(N,u)} \to \mathbb{N}$ defined as follows: for places we let $\delta(\langle a, \emptyset, k\rangle) = 0$ and $\delta(\langle a, \{e\}, k\rangle) = \delta(e)$, while for transitions we let $\delta(\langle t, H\rangle) = 1 + \delta(H)$, where $\delta(H) = \max\{\delta(x) \mid x \in H\}$. A set of elements Θ is a cut if $\mathbf{co}(\Theta)$ and there is no element x such that $\mathbf{co}(\Theta \cup \{x\})$; it is maximal if there is no $x \in \Theta$ and y not conflicting with elements in Θ such that $x \preceq y$.

$$\frac{u(a) = n,\ 1 \leq k \leq n}{\langle a, \emptyset, k\rangle \in S_{\mathcal{U}(N,u)}}$$

$$\frac{t : \bigoplus_{i\in I} a_i \to \bigoplus_j n_j b_j \in T_N, \quad \Theta = \{\langle a_i, H_i, k_i\rangle \mid i \in I\} \subseteq S_{\mathcal{U}(N,u)}, \quad \mathbf{co}(\Theta)}{e = \langle t, \Theta\rangle \in T_{\mathcal{U}(N,u)},\ \Upsilon = \{\langle b_j, \{e\}, k\rangle \mid j \in J,\ 1 \leq k \leq n_j\} \subseteq S_{\mathcal{U}(N,u)},\ \mathrm{pre}(e) = \Theta,\ \mathrm{post}(e) = \Upsilon}$$

Table 1. The unfolding $\mathcal{U}(N, u)$.

We shall use the unfolding to distill a global contract binding the interaction of participants at run-time.

Zero-safe nets. Zero-Safe nets (ZS nets) [1] are a transactional variation of PT nets, where the set of places is partitioned in two sets, of Zero-Safe places Z and stable places L, respectively. Stable markings (i.e., multisets of stable places), describe the observable states of the system, whereas the presence of tokens in Zero-Safe places denotes a marking as transient, i.e., internal to a transaction segment. The firing rule is the usual ones, except for the fact that all stable tokens produced during the transactions are made available only at the end of the transaction, when all zero-safe tokens have been consumed.

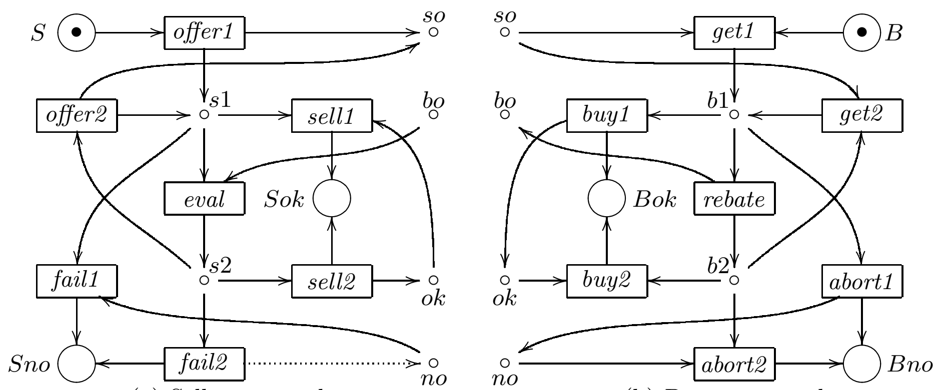
The corresponding firing rules can be explained as follow. Given a ZS net B, take the PT net $\hat{B}$ obtained from B by introducing primed version a' of all stable places a and by renaming the flow relation so to produce tokens in primed places instead of in the corresponding original stable places. More precisely, let $\hat{B}$ such that $S_{\hat{B}} = S_B \cup \{a' \mid a \in L_B\}$, $T_{\hat{B}} = T_B$ and $F_{\hat{B}}(x, y) = 0$ and $F_{\hat{B}}(x, y') = F_B(x, y)$ if $y \in L_B$, while $F_{\hat{B}}(x, y) = F_B(x, y)$ otherwise. Then a firing sequence of $\hat{B}$ is called a transaction fragment, a transaction is any firing sequence from a marking $u \in L_B^{\oplus}$ to $v \in (S_{\hat{B}} \setminus Z_B)^{\oplus}$, and its commit transforms all the primed tokens in v to ordinary tokens, i.e., leads from v to $\hat{v}$ such that $\hat{v}(x) = v(x) + v(x')$ for all $x \in L_B$ and $\hat{v}(z) = v(z)(= 0)$ otherwise.

We shall exploit ZS nets to model the interaction protocols of participants, where zero-safe places correspond to intermediate states and the protocol is implicitly concluded when only stable tokens are around.

Workflow nets. Similar ideas appear in workflow nets [4], that have a distinguished start place a_{start} with no incoming arc and a distinguished end place a_{end} with no exiting arc, playing the role of stable places in ZS nets. In fact, a workflow net is *weakly sound* if any computation that starts with a token in the start place can always terminate reaching the marking with a unique token in a_{end}. Again, all the places different from a_{start} and a_{end} are considered as modelling internal, intermediate states. We shall generalise the notion of sound workflow nets to the case where many start and end places are present and where the soundness if relative to a suitable notion of execution contract.

3 Open transactions

In this section we illustrate our approach according to the NCE scheme.



(a) Seller protocol (b) Buyer protocol

Fig. 1. Two local nets

3.1 NCE1: negotiation

The negotiation phase takes into account the available options of each participant to distill a global interaction protocol with behavioural guarantees for all.

Local and global Zero-Safe nets. The basic idea is to model interaction protocols as ZS nets: each participant describes its possible behaviour as a Zero-Safe net B_i, together with a stable marking u_i. We call these nets *local*. Each local net asserts that starting from the stable marking u_i a correct interaction should be guaranteed to lead to some other stable marking. Each local transaction is *open* in the sense that it may require some exchange of tokens with other local nets (typically, via zero-safe places only).

Given n local ZS nets such that their set of transitions are pairwise disjoint, a *global* ZS net $B = \bigcup_{i=1}^{n} B_i$ is then obtained as the union of all B_i's, with initial stable marking $u = \bigoplus_{i=1}^{n} u_i$. Typically, the local nets will share just certain zero-safe places, that are used to coordinate the local choices of participants.

Alike sound workflow nets, the ideal situation would be that starting from u any computation in $\hat{B}$ would eventually lead to a stable marking: this way we would be guaranteed that no matter which local choice is performed, no faulted situation can arise that leaves pending tokens in zero-safe places. Unfortunately, this is a too strong requirement, unrealistic in most situations, because local nets are developed according to different needs and separately from the others.

For example, consider a simple two-party situation, with a seller and a buyer whose local nets are in Fig. 1 (circles are places, smaller if zero-safe; boxes are transitions; arcs model the flow; bullets are tokens). Their interaction begins with an offer from the seller. Both the buyer and the seller may accept the last offer proposed by the other, or make a different offer, or abort the negotiation. The global net $S \cup B$ is well-behaving, according to the criteria explained above, because each transaction can eventually lead either to the marking $Sok' \oplus Bok'$ or to the marking $Sno' \oplus Bno'$. On the other hand, the transaction might not terminate if the strategy of both parties is to make offers repeatedly.

Now suppose a different seller protocol is taken, where the local decision of abandoning the negotiation is not communicated to the buyer (i.e., the arc $(\mathit{fail2}, \mathit{no})$ is removed, whence the use of a dotted line in Fig. 1). Then the global net might present erroneous transaction segments, that cannot be completed, like the firing sequence *offer1*, *get1*, *rebate*, *eval*, *fail2* that leads to the marking $Sno' \oplus b2$, which is deadlock and not stable.

Contract unfolding. The unfolding of the global net $\hat{B}$ gives a complete view of the possible interactions that can take place. In particular, the above example illustrates two typical symptoms of problematic execution, easily generalised to: 1) unfolding is not finite, 2) the unfolding exposes non stable deadlocks. The solution we propose is to model contracts as partial unfoldings, i.e. finite non-deterministic processes that satisfy some additional constraints.

The first issue can be dealt with just by assuming some bound d on the depth of elements considered in the unfolding. The bound can be enforced by adding the condition $\delta(\Theta) \leq d$ to the second rule in Table 1. Another possibility is to specify a different depth bound d_t for each transition t, in which case the condition to be added in Table 1 is $\delta(\Theta) \leq d_t$. The bound d itself can emerge, e.g., as the conjunction of the bound required by each participant when declaring their interest in the negotiation, which amounts to some sort of time-out for carrying out the whole execution. We name this property *depth boundedness*.

The second issue can be dealt with by pruning those events that may stall the execution. Formally, this can be characterised by requiring that any maximal cut contains no zero-safe place. We name this property *stability*.

A *global contract net* K is a non-empty, depth-bound and stable subnet of $\mathcal{U}(\hat{B}, u)$, i.e. a particular non-deterministic process of $(\hat{B}, u)$.

Contract pruning. When stability is violated, we can of course apply some pruning to remove those elements that cause troubles. Let Θ be a maximal cut such that $x = \langle z, \{e\}, k\rangle \in \Theta$ for some zero-safe place z, then we remove e and all y such that $e \preceq y$ (thus also x is removed) from the already partial unfolding and then iterate the pruning on the result. Clearly, as the initially considered partial unfolding is finite and at least two elements are removed at each step of the pruning algorithm, then the pruning algorithm terminates. Moreover, its result enjoys stability by definition. However, the algorithm is not always confluent. Consider the net in Fig. 2. The only problematic cut is $\{z_1, z_3\}$: if we decide to remove z_1 and therefore e_1, then we get a global contract net and we are done; viceversa if we decide to remove z_3, then also e_3, e_4 and c are removed and the resulting net has two maximal cuts $\{z_1\}$ and $\{z_2\}$ that in turn require the removal of e_1 and e_2 and we are left with the idle contract.

The algorithm can be made confluent by removing at each single step all available candidates. We call this strategy *drastic pruning*. In the example above,

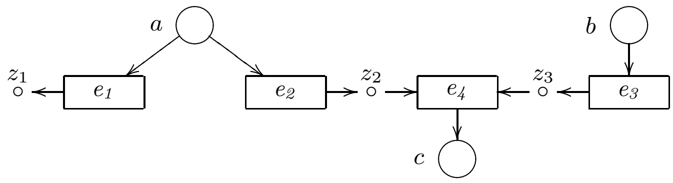


Fig. 2. A non-stable partial unfolding

e_1 and e_3 would have been removed at the first step (together with all their descendants), and e_2 at the second step, leaving again the empty contract.

Figure 3(a) shows the depth-bound unfolding of our running example (in the absence of arc $(\mathit{fail2}, \mathit{no})$), up to depth 7. It is not a global contract net, because it is not stable. In fact it has seven maximal cuts, but only three of them are stable: 1) $\{\mathit{Sok1}, \mathit{Bok1}\}$, 2) $\{\mathit{Sno1}, \mathit{Bno1}\}$, 3) $\{\mathit{Sok2}, \mathit{Bok2}\}$, 4) $\{\mathit{Sno2}, \mathit{b21}\}$, 5) $\{\mathit{s12}, \mathit{ok3}, \mathit{Bok3}\}$, 6) $\{\mathit{s12}, \mathit{no2}, \mathit{Bno2}\}$ and 7) $\{\mathit{s12}, \mathit{bo2}, \mathit{b22}\}$. The global contract net produced by drastic pruning algorithm is in Fig. 3(b).

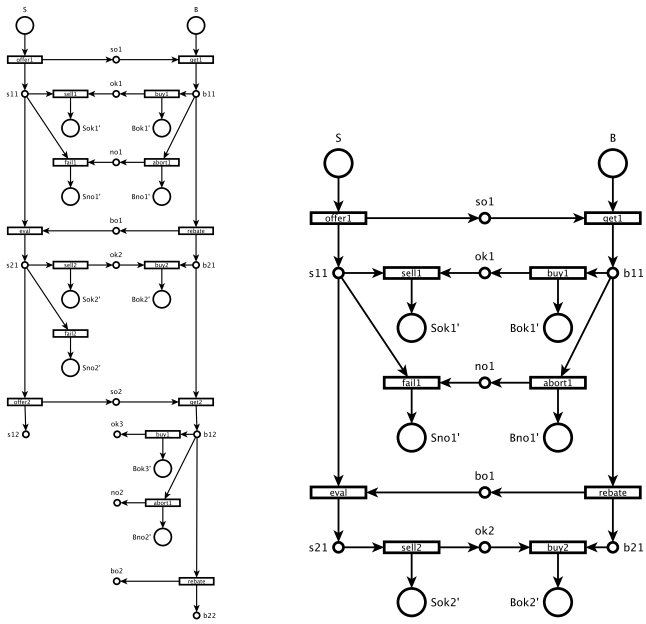
3.2 NCE2: Commit

The global contract net K can then be used as some sort of typing information that can be attached as a tag to the tokens and used in a prescriptive way during the execution phase. Hence, if the global contract net is considered viable from all participants, then it is attached as a decoration to each of the tokens in u, together with a fresh session identifier σ (the same for all tokens in u) and with the name of the token as given in K. In our running example, the two tokens in places S and B will thus be typed as $\langle \sigma, K, \langle S, \emptyset, 1\rangle\rangle$ and $\langle \sigma, K, \langle B, \emptyset, 1\rangle\rangle$, respectively. The typing information will be needed during the execution to guarantee that the contract will not be violated. We remark that in general it is not necessary to record the whole K in all tokens, but for each token x it would suffice to store the subnet of K consisting of all elements y such that $x \preceq y$.

Of course, if the global contract K is not acceptable for some participants, or if there are not enough stable tokens available, then the negotiation is considered aborted, and a new attempt has to be made. For example, the global contract net in Fig. 3(b) would constrain the seller to accept the possible rebate of the buyer, if any, which is disputable. Instead a global contract net that exclude the rebate would likely be a more convenient option.

3.3 NCE3: Execution

The final step of our approach consists in constraining the firing rules of the executable nets of all participants in order to inspect and respect the type information. Note that the executable net E_i of the ith participant can be larger than the local nets B_i exposed in the negotiation, i.e. it may contain other places and transitions and exhibit a higher degree of non-determinism without compromising the correct execution of the contract.



(a) Depth-bound (b) Global contract

Fig. 3. Building a global contract net

Tagged firing. Let E be the executable ZS net of a participant and suppose that a transition $t : \bigoplus_{i=1}^{n} a_i \rightarrow \bigoplus_{j \in J} n_j b_j \in E$ (possibly involving zero-safe places) is currently enabled and ready to consume the tagged tokens $a_1 : \tau_1, ..., a_n : \tau_n$, where $\tau_i = \langle \sigma_i, K_i, x_i \rangle$ is the type information attached to the ith token. Then t can fire if: 1) for any pair of indexes $i, i' \in [1, n]$ we have that $\sigma_i = \sigma_{i'}$; 2) each K_i contains an event $e = \langle t, \Theta \rangle$, where $\Theta = \{x_i \mid i \in [1, n]\}$. Without loss of generality, we denote by σ and K the common session identifier and global contract net of the above tokens. The firing of t will then produce the multiset of stable, non-decorated tokens $\Upsilon' = \{n_j b_j \mid j \in J, b_j \text{ is stable}\}$ and the set of tagged tokens $\Upsilon = \{b_j : \langle \sigma, K, \langle b_j, \{e\}, k \rangle \rangle \mid j \in J, b_j \text{ is zero-safe}, 1 \leq k \leq n_j\}$. Note that stable tokens are released non atomically by E, but this is not important, because we are guaranteed that the transaction will end in a finite amount of time and with no zero-safe token left. In fact, conly the choices that have been accounted for in K can be realised, and the properties of K guarantee that the overall execution is sound, no matter which local choices are taken by each participant. Finally, we remark that not all events in K must take place, because in general K can be a non-deterministic net.

4 Concluding remarks

We have proposed a net-based model for multi-party open transactions developed according to the NCE scheme. Its main features are the dynamic stipulation of contracts, the guaranteed execution and the original mix of session-based interactions and transactional distributed activities. The model is based entirely on Zero-Safe nets and the negotiated global contract is a particular non-deterministic process that can be automatically distilled from the local protocol specifications of participants. This is better suited than, say, a deterministic process, because run-time choices are unavoidable in open transactions. It is also preferable to a soundness check over the conjunction of protocols, because the check would impose too strong compatibility requirements (deemed to fail in most cases) and would not guarantee termination (and consequently would require a distributed commit phase after the execution of the transaction).

A problem that remains open is to find suitable metrics for evaluating and comparing pruning strategies different from the (confluent) drastic algorithm assumed here, which can be unnecessarily restrictive in many cases. It also remains to be investigated how our approach can be amalgamated with the recent strand of proposals that exploit workflow nets or process calculi to model various kind of compatibility notions and global / local contracts: due to space limitation we give here just a few pointers to related work for the interested reader [4–11].

References

1. Bruni, R., Montanari, U.: Zero-safe nets: Comparing the collective and individual token approaches. Inf. Comput. **156**(1-2) (2000) 46–89
2. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part i. Theor. Comput. Sci. **13** (1981) 85–108
3. Meseguer, J., Montanari, U., Sassone, V.: Process versus unfolding semantics for place/transition petri nets. Theor. Comput. Sci. **153**(1&2) (1996) 171–210
4. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. BPM'00. Vol. 1806 of LNCS, Springer (2000) 161–183
5. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. ESOP'07. Vol. 4421 of LNCS, Springer (2007) 2–17
6. Laneve, C., Padovani, L.: The pairing of contracts and session types. Concurrency, Graphs and Models. Vol. 5065 of LNCS, Springer (2008) 681–700
7. Bruni, R., Mezzina, L.G.: Types and deadlock freedom in a calculus of services, sessions and pipelines. AMAST'08. Vol. 5140 of LNCS, Springer (2008) 100–115
8. Bettini, L., Coppo, M., D'Antoni, L., Luca, M.D., Dezani-Ciancaglini, M., Yoshida, N.: Global progress in dynamically interleaved multiparty sessions. CONCUR'08. Vol. 5201 of LNCS, Springer (2008) 418–433
9. Bravetti, M., Zavattaro, G.: Contract-based discovery and composition of web services. SFM'09. Vol. 5569 of LNCS, Springer (2009) 261–295
10. Castagna, G., Gesbert, N., Padovani, L.: A theory of contracts for web services. ACM Trans. Program. Lang. Syst. **31**(5) (2009)
11. Vasconcelos, V.T.: Fundamentals of session types. SFM'09. Vol. 5569 of LNCS, Springer (2009) 158–186