AN ALGEBRA OF TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

On the theoretic analyses on transforms, they mainly focus on the computability analysis, such the Turing-completeness of unbounded precision transformers, the advantage of parallelizability of unbounded precision transformers and the limitations of bounded precision transformers. This paper give transformers an algebra for reasoning, which relies heavily on the functional decomposition results of transformers. This algebra uses functional decompositions as atomic actions, includes sequential composition and parallel composition, recursion for feedback and abstraction for hidden layers. As we know, it is the first attempt to model transformers logically and can be used to verify the correctness of transformer based applications.

1 INTRODUCTION

Because of the parallelizability of transformers Vaswani et al. (2023), transformers are becoming the mainstream techniques in large language models (LLMs). On the theoretic analyses on transforms, they mainly focus on the computability analysis, such the Turing-completeness of unbounded precision transformers Pérez et al. (2021), the advantage of parallelizability of unbounded precision transformers C. Sanford & Telgarsky (2024) and the limitations of bounded precision transformers Merrill & Sabharwal.

This paper give transformers an algebra for reasoning, which relies heavily on the functional decomposition results of transformers Yang et al. (2025), we follow its notations and definitions of transformers.

We write $[n]$ for the set $\{1, \cdots, n\}$, X^* for the set of sequences over set X, X^+ for the set of non-empty sequences over X, string Σ^* as $w = w_1 \cdots w_n$ for a finite alphabet Σ, $\mathbf{0}$ for zero vector or matrix, $\mathbf{I}$ for identity matrix, $\mathbf{1}$ for a vector or matrix whose entries are all 1's. For any function $f : \mathbb{R} \to \mathbb{R}$ and any vector $\mathbf{x} = [x_i]_{i=1}^d \in \mathbb{R}^d$, f is extended coordinate-wise:

$$f(\mathbf{x}) = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right) = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_d) \end{bmatrix}$$

We write $f : X^+ \xrightarrow{lp} Y^+$ for a function from X^+ to Y^+, which is length-preserving. And we write $f.\mathsf{a}$ for a parameter a of f.

Definition 1.1 (Transformers)**.** *A transformer is a function* $\mathsf{tf} : \Sigma^+ \xrightarrow{lp} \mathcal{Y}^+$, $\mathsf{tf}(x_1 \cdots x_n) = y_1 \cdots y_n$, *where* $\mathbf{z}_i^{(0)} = \mathsf{tf.we}(x_i) + \mathsf{tf.pe}_n(i)$, $(\mathbf{z}_1^{(1)}, \cdots, \mathbf{z}_n^{(1)}) = \mathsf{tf.tl}^{(1)}(\mathbf{z}_1^{(0)}, \cdots, \mathbf{z}_n^{(0)})$, $\cdots$, $(\mathbf{z}_1^{(L)}, \cdots, \mathbf{z}_n^{(L)}) = \mathsf{tf.tl}^{(L)}(\mathbf{z}_1^{(L-1)}, \cdots, \mathbf{z}_n^{(L-1)})$, $\mathbf{y}_i = \mathsf{tf.out}(\mathbf{z}_i^{(L)})$. *Where* $\mathsf{tf.we}$ *and* $\mathsf{tf.pe}_n$ *are word embedding function and position embedding function,* $\mathsf{tf.tl}^{(\ell)}$ *are transformer layers,* $\mathsf{tf.out}$ *is an unembedding function with some output space* $\mathcal{Y}$*. For the details of these functions, please refer to Yang et al. (2025).*

The decomposition of transformers coming from Yang et al. (2025) is shown in table 1. The idea of decomposition of a transformer is dividing a transformer into a set of atomic constructions, which are operations of vectors or matrices. For the details of these constructions, please refer to Yang et al. (2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Decomposition of Transformers

Names	Functions	Types
α_1	word embedding	$\Sigma \to \mathbb{R}^d$
α_2	position embedding	$\mathbb{N} \to \mathbb{R}^d$
α_3	identity	$\mathbb{R}^d \to \mathbb{R}^d$
α_4	min/max	$\mathbb{R}^2 \to \mathbb{R}$
α_5	add/subtract	$\mathbb{R}^2 \to \mathbb{R}$
α_6	multiply by c	$\mathbb{R} \to \mathbb{R}$
α_7	multiply xy	$\mathbb{R}^2 \to \mathbb{R}$
α_8	comparators	$\mathbb{R} \to \{0,1\}$
α_9	boolean functions	$\{0,1\}^m \to \{0,1\}$
α_{10}	continuous piecewise linear functions	$\mathbb{R}^d \to \mathbb{R}$
α_{11}	cancel residual	$\mathbb{R}^d \to \mathbb{R}^d$
α_{12}	identity	$(\mathbb{R}^d)^+ \to (\mathbb{R}^d)^+$
α_{13}	uniform average	$(\mathbb{R}^d)^+ \to (\mathbb{R}^d)^+$
α_{14}	first	$(\mathbb{R}^d)^+ \to (\mathbb{R}^d)^+$
α_{15}	index lookup	$(\mathbb{R}^d)^+ \to \mathbb{R}^+$
α_{16}	predecessor	$(\mathbb{R}^d)^+ \to (\mathbb{R}^d)^+$
α_{17}	tie-breaking	$--$
α_{18}	multi-head	$--$
α_{19}	unembedding	$--$
α_{20}	layer normalization	$\mathbb{R}^d \to \mathbb{R}^d$
ϕ_1	conditional	$\{0,1\} \times \mathbb{R}^2 \to \mathbb{R}$

end of Decomposition of Transformers

Note that, the atomic constructions include two kinds: one for atomic actions, the other for guards in programming language. The set of atomic actions is denoted $\mathsf{Act} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\} \cup \{\alpha_6, \alpha_7, \alpha_8, \alpha_9, \alpha_{10}, \alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}, \alpha_{16}, \alpha_{17}, \alpha_{18}, \alpha_{19}, \alpha_{20}\}$ and the set of guards is denoted $\mathsf{Grd} = \{\phi_1\}$. In the boolean functions, the truth value $\mathbf{true}$ is interpreted as number 1 and $\mathbf{false}$ as 0; in the general vectors or matrices operations, $\mathbf{true}$ is interpreted as $\mathbf{I}$ and $\mathbf{false}$ as $\mathbf{0}$.

2 PRELIMINARIES

For self-satisfactory, in this chapter, we introduce the preliminaries on multiset in section 2.1, equational logic in section 2.2 and pomset labelled transition systems in section 2.3.

2.1 MULTISET

Definition 2.1 (Labelled poset)**.** *A labelled poset is a tuple* $\mathbf{u} = \langle S, \leq, \lambda \rangle$*, where* S *is the carrier set,* $\leq$ *is a partial order on* S *and* λ *is a labelling function* $\lambda : S \to \Sigma$*.*

For a labelled poset $\mathbf{u}$*,* $S_{\mathbf{u}}$*,* $\leq_{\mathbf{u}}$ *and* $\lambda_{\mathbf{u}}$ *denote the carrier, the partial order and the labelling of* $\mathbf{u}$ *respectively. The set of labelled posets is denoted* LP *and the empty labelled poset is* 1*.*

Definition 2.2 (Poset morphism)**.** *For posets* $\langle A, \leq \rangle$ *and* $\langle A', \leq' \rangle$ *and function* $\varphi : A \to A'$*,* f *is called a poset morphism if for* $a_0, a_1 \in A$ *with* $a_0 \leq a_1$*, then* $\varphi(a_0) \leq' \varphi(a_1)$ *holds.*

Definition 2.3 (Labelled poset isomorphism)**.** *Let* $\mathbf{u} = \langle S_1, \leq_1, \lambda_1 \rangle$ *and* $\mathbf{v} = \langle S_2, \leq_2, \lambda_2 \rangle$ *be labelled posets. A labelled poset morphism* φ *from* $\mathbf{u} = \langle S_1, \leq_1, \lambda_1 \rangle$ *to* $\mathbf{v} = \langle S_2, \leq_2, \lambda_2 \rangle$ *is a poset morphism from* $\langle S_1, \leq_1 \rangle$ *and* $\langle S_2, \leq_2 \rangle$ *with* $\lambda_2 \circ \varphi = \lambda_1$*. Moreover,* φ *is a labelled poset isomorphism if it is a bijection with* φ^{-1} *is a poset isomorphism from* $\langle S_2, \leq_2, \lambda_2 \rangle$ *to* $\langle S_1, \leq_1, \lambda_1 \rangle$*. We say that* $\mathbf{u} = \langle S_1, \leq_1, \lambda_1 \rangle$ *is isomorphic to* $\mathbf{v} = \langle S_2, \leq_2, \lambda_2 \rangle$ *denoted* $\langle S_1, \leq_1, \lambda_1 \rangle \sim \langle S_2, \leq_2, \lambda_2 \rangle$*, if there exists a poset isomorphism* φ *between* $\langle S_1, \leq_1, \lambda_1 \rangle$ *and* $\langle S_2, \leq_2, \lambda_2 \rangle$*.*

Definition 2.4 (Multiset)**.** *A multiset is a kind of set of objects which may be repetitive denoted* $\{\!|-|\!\}$*, such that* $\{\!|a, b, b|\!\}$ *is significantly distinguishable from* $\{\!|a, b|\!\}$*.*

Definition 2.5 (Pomset). *A partially ordered multiset, pomset, is a $\sim$-equivalence class of posets. The set of pomsets is denoted* Pom*; the empty pomset is denoted* $\mathbf{I}$ *and the $\sim$-equivalence class of* $\mathbf{I}$ *is also denoted by* $\mathbf{I}$*.*

Definition 2.6 (Pomset composition in parallel). *Let $U, V \in$ Pom with $U = [\mathbf{u}]$ and $V = [\mathbf{v}]$. We write $U \parallel V$ for the parallel composition of U and V, which is the pomset represented by $\mathbf{u} \parallel \mathbf{v}$, where*

$$A_{\mathbf{u}\parallel\mathbf{v}} = A_{\mathbf{u}} \cup A_{\mathbf{v}} \qquad \leq_{\mathbf{u}\parallel\mathbf{v}} = \leq_{\mathbf{u}} \cup \leq_{\mathbf{v}}$$

Definition 2.7 (Pomset composition in sequence). *Let $U, V \in$ Pom with $U = [\mathbf{u}]$ and $V = [\mathbf{v}]$. We write $U \cdot V$ for the sequential composition of U and V, which is the pomset represented by $\mathbf{u} \cdot \mathbf{v}$, where*

$$A_{\mathbf{u}\cdot\mathbf{v}} = A_{\mathbf{u}} \cup A_{\mathbf{v}} \qquad \leq_{\mathbf{u}\cdot\mathbf{v}} = \leq_{\mathbf{u}} \cup \leq_{\mathbf{v}} \cup (A_{\mathbf{u}} \times A_{\mathbf{v}})$$

Definition 2.8 (Pomset types). *Let $U \in$ Pom, U is sequential (resp. parallel) if there exist non-empty pomsets U_1 and U_2 such that $U = U_1 \cdot U_2$ (resp. $U = U_1 \parallel U_2$).*

Definition 2.9 (Factorization). *Let $U \in$ Pom. (1) When $U = U_1 \cdot \cdots \cdot U_i \cdot \cdots \cdot U_n$ with each U_i non-sequential and non-empty, the sequence $U_1, \cdots, U_i, \cdots, U_n$ is called a sequential factorization of U. (2) When $U = U_1 \parallel \cdots \parallel U_i \parallel \cdots \parallel U_n$ with each U_i non-parallel and non-empty, the multiset $\{\!|U_1, \cdots, U_i, \cdots, U_n|\!\}$ is called a parallel factorization of U.*

Lemma 2.10 (Factorization). *Sequential and parallel factorizations exist uniquely.*

2.2 Equational Logic

Definition 2.11 (Signature). *A signature Σ consists of a finite set of function symbols (or operators) $f, g, \cdots$, where each function symbol f has an arity $ar(f)$, being its number of arguments. A function symbol $\alpha, \beta, \gamma, \cdots$ of arity* zero *is called a constant, a function symbol of arity one is called unary, and a function symbol of arity two is called binary.*

Definition 2.12 (Term). *Let Σ be a signature. The set $\mathbb{T}(\Sigma)$ of (open) terms $s, t, u, \cdots$ over Σ is defined as the least set satisfying: (1)each variable is in $\mathbb{T}(\Sigma)$; (2) if $f \in \Sigma$ and $t_1, \cdots, t_{ar(f)} \in \mathbb{T}(\Sigma)$, then $f(t_1, \cdots, t_{ar(f)} \in \mathbb{T}(\Sigma))$. A term is closed if it does not contain free variables. The set of closed terms is denoted by $\mathcal{T}(\Sigma)$.*

Definition 2.13 (Substitution). *Let Σ be a signature. A substitution is a mapping σ from variables to the set $\mathbb{T}(\Sigma)$ of open terms. A substitution extends to a mapping from open terms to open terms: the term $\sigma(t)$ is obtained by replacing occurrences of variables x in t by $\sigma(x)$. A substitution σ is closed if $\sigma(x) \in \mathcal{T}(\Sigma)$ for all variables x.*

Definition 2.14 (Axiomatization). *An axiomatization over a signature Σ is a finite set of equations, called axioms, of the form $s = t$ with $s, t \in \mathbb{T}(\Sigma)$.*

Definition 2.15 (Equality relation). *An axiomatization over a signature Σ induces a binary equality relation $=$ on $\mathbb{T}(\Sigma)$ as follows. (1)(Substitution) If $s = t$ is an axiom and σ a substitution, then $\sigma(s) = \sigma(t)$. (2)(Equivalence) The relation $=$ is closed under reflexivity, symmetry, and transitivity. (3)(Context) The relation $=$ is closed under contexts: if $t = u$ and f is a function symbol with $ar(f) > 0$, then $f(s_1, \cdots, s_{i-1}, t, s_{i+1}, \cdots, s_{ar(f)}) = f(s_1, \cdots, s_{i-1}, u, s_{i+1}, \cdots, s_{ar(f)})$.*

Definition 2.16 (Model). *Assume an axiomatization $\mathcal{E}$ over a signature Σ, which induces an equality relation $=$. A model for $\mathcal{E}$ consists of a set $\mathcal{M}$ together with a mapping $\phi : \mathcal{T}(\Sigma) \to \mathcal{M}$. (1) $(\mathcal{M}, \phi)$ is sound for $\mathcal{E}$ if $s = t$ implies $\phi(s) \equiv \phi(t)$ for $s, t \in \mathcal{T}(\Sigma)$; (2) $(\mathcal{M}, \phi)$ is complete for $\mathcal{E}$ if $\phi(s) \equiv \phi(t)$ implies $s = t$ for $s, t \in \mathcal{T}(\Sigma)$.*

Definition 2.17 (Term rewriting system). *Assume a signature Σ. A rewrite rule is an expression $s \to t$ with $s, t \in \mathbb{T}(\Sigma)$, where: (1) the left-hand side s is not a single variable; (2) all variables that occur at the right-hand side t also occur in the left-hand side s. A term rewriting system (TRS) is a finite set of rewrite rules.*

Definition 2.18 (Rewrite relation). *A TRS over a signature Σ induces an one-step rewrite relation $\to$ on $\mathbb{T}(\Sigma)$ as follows. (1) (Substitution) If $s \to t$ is a rewrite rule and σ a substitution, then $\sigma(s) \to \sigma(t)$. (2) (Context) The relation $\to$ is closed under contexts: if $t \to u$ and f is a function symbol with $ar(f) > 0$, then $f(s_1, \cdots, s_{i-1}, t, s_{i+1}, \cdots, s_{ar(f)}) \to f(s_1, \cdots, s_{i-1}, u, s_{i+1}, \cdots, s_{ar(f)})$.*

The rewrite relation $\to^*$ is the reflexive transitive closure of the one-step rewrite relation $\to$: (1) if $s \to t$, then $s \to^* t$; (2) $t \to^* t$; (3) if $s \to^* t$ and $t \to^* u$, then $s \to^* u$.

Definition 2.19 (Normal form)**.** *A term is called a normal form for a TRS if it cannot be reduced by any of the rewrite rules.*

Definition 2.20 (Termination)**.** *A TRS is terminating if it does not induce infinite reductions $t_0 \to t_1 \to t_2 \to \cdots$.*

Definition 2.21 (Weak confluence)**.** *A TRS is weakly confluent if for each pair of one-step reductions $s \to t_1$ and $s \to t_2$, there is a term u such that $t_1 \to^* u$ and $t_2 \to^* u$.*

Theorem 2.22 (Newman's lemma)**.** *If a TRS is terminating and weakly confluent, then it reduces each term to a unique normal form.*

Definition 2.23 (Convergence)**.** *A pair of terms s and t is said to be convergent if there exists a term u such that $s \to^* u$ and $t \to^* u$.*

Axiomatizations can give rise to TRSs that are not weakly confluent, which can be remedied by Knuth-Bendix completion Knuth & Bendix (1970). It determines overlaps in left hand sides of rewrite rules, and introduces extra rewrite rules to join the resulting right hand sides, witch are called critical pairs.

Theorem 2.24 (Weak confluence)**.** *A TRS is weakly confluent if and only if all its critical pairs are convergent.*

Definition 2.25 (Elimination property)**.** *Let a process algebra with a defined set of basic terms as a subset of the set of closed terms over the process algebra. Then the process algebra has the elimination to basic terms property if for every closed term s of the algebra, there exists a basic term t of the algebra such that the algebra$\vdash s = t$.*

Definition 2.26 (Strongly normalizing)**.** *A term s_0 is called strongly normalizing if does not an infinite series of reductions beginning in s_0.*

Definition 2.27. *We write $s >_{lpo} t$ if $s \to^+ t$ where $\to^+$ is the transitive closure of the reduction relation defined by the transition rules of an algebra.*

Theorem 2.28 (Strong normalization)**.** *Let a term rewriting system (TRS) with finitely many rewriting rules and let $>$ be a well-founded ordering on the signature of the corresponding algebra. If $s >_{lpo} t$ for each rewriting rule $s \to t$ in the TRS, then the term rewriting system is strongly normalizing.*

2.3 Pomset Labelled Transition System

Definition 2.29 (Pomset labelled transition system)**.** *A pomset labelled transition system (PLTS) is a quadruple $\langle \mathsf{Proc}, \mathsf{Act}, \{\xrightarrow{U} | U \in \mathsf{Pom}\}, \mathsf{Pred}\rangle$, where:*

1. Proc *is a set of processes, ranged over by p, q.*
2. Act *is a set of actions, ranged over by a, b.*
3. Pom *is the set of pomsets over* Act*, ranged over by U, V.*
4. $\xrightarrow{U} \subseteq \mathsf{Proc} \times \mathsf{Pom} \times \mathsf{Proc}$ *is called a pomset transition for every $U \in \mathsf{Pom}$. We write $p \xrightarrow{U} q$ instead of $(p, q) \in \xrightarrow{U}$, and write $p \not\xrightarrow{U}$ if $p \xrightarrow{U} q$ with no state q. Intuitively, $p \xrightarrow{U} q$ means that state p can evolve into state q by the execution of pomset U. We see that traditional single action transition $p \xrightarrow{a} q$ with $a \in \mathsf{Act}$ is a special case of pomset transition in which the pomset is primitive.*
5. *For every $P \in \mathsf{Pred}$, we write pP (resp. $p\neg P$) if state p satisfies (resp. does not satisfy) predicate P. Intuitively, pP means that predicate P holds in state p.*

The binary pomset transitions $p \xrightarrow{U} q$ and unary predicates pP in a PLTS are called transitions.

Note that, by replacing $U \in \mathsf{Pom}$ by $a \in \mathsf{Act}$, we can get the definition of traditional labelled transition system (LTS). And we use $\mathsf{Act}(p) \subseteq \mathsf{Act}$ to denote the set of actions occurring in p.

Definition 2.30 (Finiteness conditions on a PLTS)**.** *A PLTS is:*

- *Finitely branching: if for every state p there are only finitely many outgoing pomset transitions $p \xrightarrow{U} q$.*
- *Regular: if it is finitely branching and each state can reach only finitely many other states.*
- *Finite: if it is finitely branching and there is no infinite sequence of pomset transitions $p_0 \xrightarrow{U_0} p_1 \xrightarrow{U_1} \cdots$.*

A pomset transition system specification is a collection of inductive proof rules to derive the pomset transitions over states in Proc.

Definition 2.31 (Pomset transition system specification)**.** *Let Σ be a signature, and $t, t' \in \mathbb{T}(\Sigma)$. A pomset transition rule ρ is of the form $\frac{H}{\alpha}$, where H is the set of premises with positive premises $t \xrightarrow{U} t'$ and tP, and negative premises $t \overset{U}{\not\to}$ and $t\neg P$; α is the conclusion with the form $t \xrightarrow{U} t'$ and tP, and $U \in \mathsf{Pom}$ and P is a predicate. For the conclusion with the form of $t \xrightarrow{U} t'$, the left-hand side of the conclusion t is called the source of ρ and the right-hand side of the conclusion t' is called the target of ρ. A transition rule is closed if it does not contain variables.*

A pomset transition system specification (PTSS) is a set of pomset transition rules. A PTSS is positive if its transition rules do not contain negative premises.

Definition 2.32 (Pomset transitions)**.** *Let $p, q \in \mathsf{Proc}$ and $U \in \mathsf{Pom}$, $p \xrightarrow{U} q$ is called a pomset transition from p to q.*

Definition 2.33 (Pomset bisimulation)**.** *Let $\langle \mathsf{Proc}, \mathsf{Act}, \{\xrightarrow{U} | U \in \mathsf{Pom}\}, \mathsf{Pred}\rangle$ be a PLTS, S a set of states, and $p, q \in \mathsf{Proc}$ and $s, s' \in S$. A pomset bisimulation is a relation $R \subseteq \langle \mathsf{Proc}, S\rangle \times \langle \mathsf{Proc}, S\rangle$, such that:*

1. *If $\langle p, s\rangle R\langle q, s\rangle$ and $\langle p, s\rangle \xrightarrow{U} \langle p', s'\rangle$, then $\langle q, s\rangle \xrightarrow{U} \langle q', s'\rangle$, with $U \in \mathsf{Pom}$, and $\langle p', s'\rangle R\langle q', s'\rangle$.*
2. *If $\langle p, s\rangle R\langle q, s\rangle$ and $\langle q, s\rangle \xrightarrow{U} \langle q', s'\rangle$, then $\langle p, s\rangle \xrightarrow{U} \langle p', s'\rangle$, with $U \in \mathsf{Pom}$, and $\langle p', s'\rangle R\langle q', s'\rangle$.*
3. *If $\langle p, s\rangle R\langle q, s\rangle$ and $\langle p, s\rangle P$, then $\langle q, s\rangle P$.*
4. *If $\langle p, s\rangle R\langle q, s\rangle$ and $\langle q, s\rangle P$, then $\langle p, s\rangle P$.*

We say that p, q are pomset bisimilar, written $p \underline{\leftrightarrow} q$, if there exists a pomset bisimulation R, such that $\langle p, s\rangle R\langle q, s\rangle$.

Definition 2.34 (Congruence)**.** *Let Σ be a signature. An equivalence relation R on $\mathcal{T}(\Sigma)$ is a congruence if for each $f \in \Sigma$, if $s_i R t_i$ for $i \in \{1, \cdots, ar(f)\}$, then $f(s_1, \cdots, s_{ar(f)}) R f(t_1, \cdots, t_{ar(f)})$.*

Definition 2.35 (Operational conservative extension)**.** *Σ_0 and Σ_1 are two signatures, the union of Σ_0 and Σ_1, denoted $\Sigma_0 \oplus \Sigma_1$, agrees on the arity of the function symbols in the intersection of Σ_0 and Σ_1. T_0 and T_1 are PTSSs over Σ_0 and Σ_1, the sum of T_0 and T_1, denoted $T_0 \oplus T_1$, is the PTSS over signature $\Sigma_0 \oplus \Sigma_1$ containing the pomset transition rules in T_0 and T_1. A PTSS $T_0 \oplus T_1$ is an operational conservative extension of PTSS T_0, if for each closed transition rule $\frac{N}{\alpha}$ such that:*

- *N contains only negative literals.*
- *The left-hand side of α is in $T(\Sigma_0)$.*
- *$T_0 \oplus T_1 \vdash \frac{N}{\alpha}$.*

we have that $T_0 \vdash \frac{N}{\alpha}$.

Definition 2.36 (Source-dependency)**.** *The source-dependent variables in a pomset transition rule ρ are defined inductively as follows:*

- *All variables in the source of ρ are source-dependent.*
- *If $t \xrightarrow{U} t'$ is a premise of ρ and all variables in t are source-dependent, then all variables in t' are source-dependent.*

A pomset transition rule is source-dependent if all its variables are.

Definition 2.37 (Freshness)**.** *We say that a term in $\mathbb{T}(\Sigma_0 \oplus \Sigma_1)$ is fresh if it contains a function symbol (may be a constant) from $\Sigma_1 \setminus \Sigma_0$. Similarly, a predicate symbol in T_1 is fresh if it does not occur in T_0.*

Theorem 2.38 (Operational conservative extension)**.** *Let T_0 and T_1 be PTSSs over signature Σ_0 and Σ_1, respectively. $T_0 \oplus T_1$ is an operational conservative extension of T_0, if:*

1. *Each $\rho \in T_0$ is source-dependent.*
2. *For each $\rho \in T_1$,*
   - *Either the source of ρ is fresh.*
   - *Or ρ has a premise of the form $t \xrightarrow{U} t'$ or tP, where:*
     - *$t \in \mathbb{T}(\Sigma_0)$.*
     - *All variables in t occur in the source of ρ.*
     - *t', P and every $a \in U$ are fresh.*

3 Basic Algebra for Transformers

In this section, we will discuss the algebraic laws for the set of atomic actions Act the set of guards Grd, the deadlock constant $\mathbf{0}$ and the identity $\mathbf{I}$, the set of pomsets over $\mathsf{Act} \cup \{\mathbf{0}\} \cup \{\mathbf{I}\}$ denoted Pom, and the serial composition $\circ$ and the alternative composition $+$, and let $\alpha \in \mathsf{Act} \cup \{\mathbf{0}\} \cup \{\mathbf{I}\}$, $U \in \mathsf{Pom}$. The resulted algebra is called Basic Algebra for True Concurrency, abbreviated BAT.

3.1 Axiomatic System

In the following, let $\alpha, \beta, \alpha', \beta' \in \mathsf{Act}$, $\phi, \psi \in \mathsf{Grd}$, and let variables x, y, z range over the set of terms BAT, p, q, s range over the set of closed terms. The set of axioms of BAT consists of the laws given in Table 2. The predicate $test(\phi, s)$ represents that ϕ holds in the state s, and $test(\mathbf{I}, s)$ holds and $test(\mathbf{0}, s)$ does not hold. $effect(U, s) \in S$ denotes s' in $s \xrightarrow{U} s'$ for $U \in \mathsf{Pom}$. The predicate weakest precondition $wp(U, \phi)$ denotes that $\forall s, s' \in S, test(\phi, effect(U, s))$ holds.

We extend Grd to the set of basic guards G with element $\phi, \psi, \cdots$, which is generated by the following formation rules:

$$\phi ::= \mathbf{0} \mid \mathbf{I} \mid \neg\phi \mid \psi \in \mathsf{Grd} \mid \phi + \psi \mid \phi \circ \psi$$

The deadlock $\mathbf{0}$ is caused by incomposibility of serial composition, which may be caused by the mismatch of dimensions of vectors or matrices, we define the composibility function of two actions $a, b \in \mathsf{Act}$ as follows:

$$\gamma(a, b) = \begin{cases} a \circ b, & a, b \text{ are composible;} \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Each serial composition implicitly executes this function.

Table 2: Axioms of BAT

No.	Axioms
$A1$	$x + y = y + x$
$A2$	$(x + y) + z = x + (y + z)$

to be continued...

continuing: Axioms of BAT

No.	Axioms
$A3$	$x + x = x$
$A4$	$(x + y) \circ z = x \circ z + y \circ z$
$A5$	$(x \circ y) \circ z = x \circ (y \circ z)$
$A6$	$x + \mathbf{0} = x$
$A7$	$\mathbf{0} \circ x = \mathbf{0}$
$A8$	$\mathbf{I} \circ x = x$
$A9$	$x \circ \mathbf{I} = x$
$G1$	$\phi \circ \neg\phi = \mathbf{0}$
$G2$	$\phi + \neg\phi = \mathbf{I}$
$G3$	$\phi \circ \mathbf{0} = \mathbf{0}$
$G4$	$\phi \circ (x + y) = \phi \circ x + \phi \circ y$
$G5$	$\phi \circ (x \circ y) = (\phi \circ x) \circ y$
$G6$	$(\phi + \psi) \circ x = \phi \circ x + \psi \circ x$
$G7$	$(\phi \circ \psi) \circ x = \phi \circ (\psi \circ x)$
$G8$	$\phi = \mathbf{I}$ if $\forall s \in S. test(\phi, s)$
$G9$	$\phi_0 \circ \cdots \circ \phi_n = \mathbf{0}$ if $\forall s \in S, \exists i \leq n. test(\neg\phi_i, s)$
$G10$	$wp(U, \phi)U\phi = wp(U, \phi)U$
$G11$	$\neg wp(U, \phi)U\neg\phi = \neg wp(U, \phi)U$

end of Axioms of BAT

Definition 3.1 (Basic terms of BAT)**.** *The set of basic terms of BAT, $\mathcal{B}(BAT)$, is inductively defined as follows:*

1. $\mathsf{Act} \cup \{\mathbf{0}\} \cup \{\mathbf{I}\} \subset \mathcal{B}(BAT)$.
2. $\mathbf{Grd} \subset \mathcal{B}(BAT)$.
3. *If* $U \in \mathsf{Pom}, t \in \mathcal{B}(BAT)$ *then* $U \circ t \in \mathcal{B}(BAT)$.
4. *If* $\phi \in \mathsf{Grd}, t \in \mathcal{B}(BAT)$ *then* $\phi \circ t \in \mathcal{B}(BAT)$.
5. *If* $t, s \in \mathcal{B}(BAT)$ *then* $t + s \in \mathcal{B}(BAT)$.

Theorem 3.2 (Elimination theorem of BAT)**.** *Let p be a closed BAT term. Then there is a basic BAT term q such that $BAT \vdash p = q$.*

Proof. (1) Firstly, suppose that the following ordering on the signature of BAT is defined: $\circ > +$ and the symbol $\circ$ is given the lexicographical status for the first argument, then for each rewrite rule $p \to q$ in Table 3 relation $p >_{lpo} q$ can easily be proved. We obtain that the term rewrite system shown in Table 3 is strongly normalizing, for it has finitely many rewriting rules, and $>$ is a well-founded ordering on the signature of BAT, and if $s >_{lpo} t$, for each rewriting rule $s \to t$ is in Table 3 (see Theorem 2.28).

Table 3: Rewriting rules of BAT

No.	Rewriting rules
$RA3$	$x + x \to x$
$RA4$	$(x + y) \circ z \to x \circ z + y \circ z$
$RA5$	$(x \circ y) \circ z \to x \circ (y \circ z)$
$RA6$	$x + \mathbf{0} \to x$
$RA7$	$\mathbf{0} \circ x \to \mathbf{0}$
$RA8$	$\mathbf{I} \circ x \to x$
$RA9$	$x \circ \mathbf{I} \to x$
$RG1$	$\phi \circ \neg\phi \to \mathbf{0}$
$RG2$	$\phi + \neg\phi \to \mathbf{I}$

to be continued...

continuing: Rewriting rules of BAT

No.	Rewriting rules
$RG3$	$\phi \circ \mathbf{0} \to \mathbf{0}$
$RG4$	$\phi \circ (x + y) \to \phi \circ x + \phi \circ y$
$RG5$	$\phi \circ (x \circ y) \to (\phi \circ x) \circ y$
$RG6$	$(\phi + \psi) \circ x \to \phi \circ x + \psi \circ x$
$RG7$	$(\phi \circ \psi) \circ x \to \phi \circ (\psi \circ x)$
$RG8$	$\phi \to \mathbf{I}$ if $\forall s \in S.test(\phi, s)$
$RG9$	$\phi_0 \circ \cdots \circ \phi_n \to \mathbf{0}$ if $\forall s \in S, \exists i \leq n.test(\neg\phi_i, s)$
$RG10$	$wp(U, \phi)U\phi \to wp(U, \phi)U$
$RG11$	$\neg wp(U, \phi)U\neg\phi \to \neg wp(U, \phi)U$

end of Axioms of BAT

(2) Then we prove that the normal forms of closed BAT terms are basic BAT terms.

Suppose that p is a normal form of some closed BAT term and suppose that p is not a basic term. Let p' denote the smallest sub-term of p which is not a basic term. It implies that each sub-term of p' is a basic term. Then we prove that p is not a term in normal form. It is sufficient to induct on the structure of p':

- Case $p' \equiv U, U \in \mathsf{Pom}$. p' is a basic term, which contradicts the assumption that p' is not a basic term, so this case should not occur.
- Case $p' \equiv \phi, \phi \in \mathsf{Grd}$. p' is a basic term, which contradicts the assumption that p' is not a basic term, so this case should not occur.
- Case $p' \equiv p_1 \circ p_2$. By induction on the structure of the basic term p_1:
  - Subcase $p_1 \in \mathsf{Pom}$. p' would be a basic term, which contradicts the assumption that p' is not a basic term;
  - Subcase $p_1 \in \mathsf{Grd}$. p' would be a basic term, which contradicts the assumption that p' is not a basic term;
  - Subcase $p_1 \equiv U \circ p_1'$. $RA5$ or $RA9$ rewriting rule can be applied. So p is not a normal form;
  - Subcase $p_1 \equiv \phi \circ p_1'$. $RG1$, $RG3$, $RG4$, $RG5$, $RG7$, or $RG8 - 9$ rewriting rules can be applied. So p is not a normal form;
  - Subcase $p_1 \equiv p_1' + p_1''$. $RA4$, $RA6$, $RG2$, or $RG6$ rewriting rules can be applied. So p is not a normal form.
- Case $p' \equiv p_1 + p_2$. By induction on the structure of the basic terms both p_1 and p_2, all subcases will lead to that p' would be a basic term, which contradicts the assumption that p' is not a basic term.

□

3.2 Operational Semantics

We give the operational transition rules for $\mathbf{I}$, atomic action $\alpha \in \mathsf{Act}$, pomset $U \in \mathsf{Pom}$, guard $\phi \in \mathsf{Grd}$ and operators $\circ$ and $+$ as Table 4 shows. And the predicate $\xrightarrow{U} \surd$ represents successful termination after execution of U.

Theorem 3.3 (Congruence of BAT with respect to pomset bisimulation equivalence)**.** *Pomset bisimulation equivalence is a congruence with respect to BAT.*

Proof. Since the PTSS of BAT in Table 4 is positive after reduction and in panth format, pomset bisimulation equivalence is a congruence with respect to BAT. □

$$\frac{}{\langle \mathbf{I}, s\rangle \to \langle \surd, s\rangle}$$

$$\frac{}{\langle U, s\rangle \xrightarrow{U} \langle \surd, s'\rangle} \text{ if } s' \in effect(U, s)$$

$$\frac{}{\langle \phi, s\rangle \to \langle \surd, s\rangle} \text{ if } test(\phi, s)$$

$$\frac{\langle x, s\rangle \xrightarrow{U} \langle \surd, s'\rangle}{\langle x + y, s\rangle \xrightarrow{U} \langle \surd, s'\rangle} \quad \frac{\langle x, s\rangle \xrightarrow{U} \langle x', s'\rangle}{\langle x + y, s\rangle \xrightarrow{U} \langle x', s'\rangle}$$

$$\frac{\langle y, s\rangle \xrightarrow{U} \langle \surd, s'\rangle}{\langle x + y, s\rangle \xrightarrow{U} \langle \surd, s'\rangle} \quad \frac{\langle y, s\rangle \xrightarrow{U} \langle y', s'\rangle}{\langle x + y, s\rangle \xrightarrow{U} \langle y', s'\rangle}$$

$$\frac{\langle x, s\rangle \xrightarrow{U} \langle \surd, s'\rangle}{\langle x \circ y, s\rangle \xrightarrow{U} \langle y, s'\rangle} \quad \frac{\langle x, s\rangle \xrightarrow{U} \langle x', s'\rangle}{\langle x \circ y, s\rangle \xrightarrow{U} \langle x' \circ y, s'\rangle}$$

Table 4: Transition rules of BAT

3.3 Soundness and Completeness

Theorem 3.4 (Soundness of BAT modulo pomset bisimulation equivalence)**.** *The axiomatization of BAT is sound modulo pomset bisimulation equivalence, i.e., let x and y be BAT terms. If $BAT \vdash x = y$, then $x \underline{\leftrightarrow} y$.*

Proof. Since pomset bisimulation $\underline{\leftrightarrow}$ is both an equivalent and a congruent relation, we only need to check if each axiom in Table 2 is sound modulo pomset bisimulation equivalence $\underline{\leftrightarrow}$. The proof is trivial and we omit it. □

Theorem 3.5 (Completeness of BAT modulo pomset bisimulation equivalence)**.** *The axiomatization of BAT is complete modulo pomset bisimulation equivalence, i.e., let p and q be closed BAT terms, if $p \underline{\leftrightarrow} q$ then $p = q$.*

Proof. Firstly, by the elimination theorem of BAT, we know that for each closed BAT term p, there exists a closed basic BAT term p', such that $BAT \vdash p = p'$, so, we only need to consider closed basic BAT terms.

The basic terms (see Definition 3.1) modulo associativity and commutativity (AC) of conflict + (defined by axioms $A1$ and $A2$ in Table 2), and this equivalence is denoted by $=_{AC}$. Then, each equivalence class s modulo AC of + has the following normal form

$$s_1 + \cdots + s_k$$

with each s_i either an atomic event or of the form $t_1 \circ t_2$, and each s_i is called the summand of s.

Now, we prove that for normal forms n and n', if $n \underline{\leftrightarrow} n'$ then $n =_{AC} n'$. It is sufficient to induct on the sizes of n and n'.

- Consider a summand U of n. Then $\langle n, s\rangle \xrightarrow{U} \langle \surd, s'\rangle$, so $n \sim_p n'$ implies $\langle n', s\rangle \xrightarrow{U} \langle \surd, s\rangle$, meaning that n' also contains the summand U.
- Consider a summand ϕ of n. Then $\langle n, s\rangle \to \langle \surd, s\rangle$, if $test(\phi, s)$ holds, so $n \sim_p n'$ implies $\langle n', s\rangle \to \langle \surd, s\rangle$, if $test(\phi, s)$ holds, meaning that n' also contains the summand ϕ.
- Consider a summand $t_1 \circ t_2$ of n. Then $\langle n, s\rangle \xrightarrow{t_1} \langle t_2, s'\rangle$, so $n \sim_p n'$ implies $\langle n', s\rangle \xrightarrow{t_1} \langle t_2', s'\rangle$ with $t_2 \sim_p t_2'$, meaning that n' contains a summand $t_1 \circ t_2'$. Since t_2 and t_2' are normal forms and have sizes smaller than n and n', by the induction hypotheses $t_2 \underline{\leftrightarrow} t_2'$ implies $t_2 =_{AC} t_2'$.

$$\frac{x \xrightarrow{\alpha} \surd \quad y \xrightarrow{\alpha'} \surd}{x \oplus y \xrightarrow{\{\!|\alpha,\alpha'|\!\}} \surd} \qquad \frac{x \xrightarrow{\alpha} x' \quad y \xrightarrow{\alpha'} \surd}{x \oplus y \xrightarrow{\{\!|\alpha,\alpha'|\!\}} x'}$$

$$\frac{x \xrightarrow{\alpha} \surd \quad y \xrightarrow{\alpha'} y'}{x \oplus y \xrightarrow{\{\!|\alpha,\alpha'|\!\}} y'} \qquad \frac{x \xrightarrow{\alpha} x' \quad y \xrightarrow{\alpha'} y'}{x \oplus y \xrightarrow{\{\!|\alpha,\alpha'|\!\}} x' \oplus y'}$$

Table 5: Transition rules of APT

So, we get $n =_{AC} n'$.

Finally, let s and t be basic terms, and $s \underline{\leftrightarrow} t$, there are normal forms n and n', such that $s = n$ and $t = n'$. The soundness theorem of BAT modulo pomset bisimulation equivalence (see Theorem 3.4) yields $s \underline{\leftrightarrow} n$ and $t \underline{\leftrightarrow} n'$, so $n \underline{\leftrightarrow} s \underline{\leftrightarrow} t \underline{\leftrightarrow} n'$. Since if $n \underline{\leftrightarrow} n'$ then $n =_{AC} n'$, $s = n =_{AC} n' = t$, as desired. □

4 Algebra of Parallelism for Transformers

In this section, we will discuss parallel composition. The resulted algebra is called Algebra of Parallelism for Transformers, abbreviated APT.

4.1 Operational Semantics

Firstly, we give the transition rules for parallelism as table 5 shows.

Theorem 4.1 (Congruence theorem of APT)**.** *Pomset bisimulation equivalences $\underline{\leftrightarrow}$ is a congruence with respect to APT.*

Proof. Since the PTSS of APT in Table 5 is positive after reduction and in panth format, pomset bisimulation equivalence $\underline{\leftrightarrow}$ is a congruence with respect to APT. □

4.2 Axiomatic System

We design the axioms of APT in Table 6.

Table 6: Axioms of APT

No.	Axioms
$P1$	$\alpha \oplus (\alpha' \circ y) = (\alpha \oplus \alpha') \circ y$
$P2$	$(\alpha \circ x) \oplus \alpha' = (\alpha \oplus \alpha') \circ x$
$P3$	$(\alpha \circ x) \oplus (\alpha' \circ y) = (\alpha \oplus \alpha') \circ (x \oplus y)$
$P4$	$(x + y) \oplus z = (x \oplus z) + (y \oplus z)$
$P5$	$x \oplus (y + z) = (x \oplus y) + (x \oplus z)$
$P6$	$\mathbf{0} \oplus x = \mathbf{0}$
$P7$	$x \oplus \mathbf{0} = \mathbf{0}$
$P8$	$\mathbf{I} \oplus x = x$
$P9$	$x \oplus \mathbf{I} = x$
$G12$	$\phi \circ (x \oplus y) = \phi \circ x \oplus \phi \circ y$
$G13$	$\phi \oplus \mathbf{0} = \mathbf{0}$
$G14$	$\mathbf{0} \oplus \phi = \mathbf{0}$
$G15$	$\phi \oplus \mathbf{I} = \phi$
$G16$	$\mathbf{I} \oplus \phi = \phi$
$G17$	$\phi \oplus \neg\phi = \mathbf{0}$
$G18$	$\phi_0 \oplus \cdots \oplus \phi_n = \mathbf{0}$ if $\forall s_0, \cdots, s_n \in S, \exists i \leq n.test(\neg\phi_i, s_0 \cup \cdots \cup s_n)$

end of Axioms of APT

Definition 4.2 (Basic terms of APT)**.** *The set of basic terms of APT, $\mathcal{B}(APT)$, is inductively defined as follows:*

1. $\mathsf{Act} \cup \{\mathbf{0}\} \cup \{\mathbf{I}\} \subset \mathcal{B}(APT)$.
2. $\mathsf{Grd} \subset \mathcal{B}(APT)$.
3. *If* $U \in \mathsf{Pom}, t \in \mathcal{B}(APT)$ *then* $U \circ t \in \mathcal{B}(APT)$.
4. *If* $\phi \in \mathsf{Grd}, t \in \mathcal{B}(APT)$ *then* $\phi \circ t \in \mathcal{B}(APT)$.
5. *If* $t, s \in \mathcal{B}(APT)$ *then* $t + s \in \mathcal{B}(APT)$.
6. *If* $t, s \in \mathcal{B}(APT)$ *then* $t \oplus s \in \mathcal{B}(APT)$.

Theorem 4.3 (Elimination theorem of parallelism)**.** *Let p be a closed APT term. Then there is a basic APT term q such that $APT \vdash p = q$.*

Proof. (1) Firstly, suppose that the following ordering on the signature of APT is defined: $\oplus > \circ > +$ and the symbol $\circ$ is given the lexicographical status for the first argument, then for each rewrite rule $p \to q$ in Table 7 relation $p >_{lpo} q$ can easily be proved. We obtain that the term rewrite system shown in Table 7 is strongly normalizing, for it has finitely many rewriting rules, and $>$ is a well-founded ordering on the signature of APT, and if $s >_{lpo} t$, for each rewriting rule $s \to t$ is in Table 7 (see Theorem 2.28).

Table 7: Rewriting rules of APT

No.	Rewriting rules
$RP1$	$\alpha \oplus (\alpha' \circ y) \to (\alpha \oplus \alpha') \circ y$
$RP2$	$(\alpha \circ x) \oplus \alpha' \to (\alpha \oplus \alpha') \circ x$
$RP3$	$(\alpha \circ x) \oplus (\alpha' \circ y) \to (\alpha \oplus \alpha') \circ (x \oplus y)$
$RP4$	$(x + y) \oplus z \to (x \oplus z) + (y \oplus z)$
$RP5$	$x \oplus (y + z) \to (x \oplus y) + (x \oplus z)$
$RP6$	$\mathbf{0} \oplus x \to \mathbf{0}$
$RP7$	$x \oplus \mathbf{0} \to \mathbf{0}$
$RP8$	$\mathbf{I} \oplus x \to x$
$RP9$	$x \oplus \mathbf{I} \to x$
$RG12$	$\phi \circ (x \oplus y) \to \phi \circ x \oplus \phi \circ y$
$RG13$	$\phi \oplus \mathbf{0} \to \mathbf{0}$
$RG14$	$\mathbf{0} \oplus \phi \to \mathbf{0}$
$RG15$	$\phi \oplus \mathbf{I} \to \phi$
$RG16$	$\mathbf{I} \oplus \phi \to \phi$
$RG17$	$\phi \oplus \neg\phi \to \mathbf{0}$
$RG18$	$\phi_0 \oplus \cdots \oplus \phi_n \to \mathbf{0}$ if $\forall s_0, \cdots, s_n \in S, \exists i \leq n.test(\neg\phi_i, s_0 \cup \cdots \cup s_n)$

end of Rewriting rules of APT

(2) Then we prove that the normal forms of closed APT terms are basic APT terms.

Suppose that p is a normal form of some closed APT term and suppose that p is not a basic APT term. Let p' denote the smallest sub-term of p which is not a basic APT term. It implies that each sub-term of p' is a basic APT term. Then we prove that p is not a term in normal form. It is sufficient to induct on the structure of p':

- Case $p' \equiv U, U \in \mathsf{Pom}$. p' is a basic APT term, which contradicts the assumption that p' is not a basic APT term, so this case should not occur.

- Case $p' \equiv \phi, \phi \in \mathsf{Grd}$. p' is a basic term, which contradicts the assumption that p' is not a basic term, so this case should not occur.
- Case $p' \equiv p_1 \circ p_2$. By induction on the structure of the basic APT term p_1:
  - Subcase $p_1 \in \mathsf{Pom}$. p' would be a basic APT term, which contradicts the assumption that p' is not a basic APT term;
  - Subcase $p_1 \in \mathsf{Grd}$. p' would be a basic term, which contradicts the assumption that p' is not a basic term;
  - Subcase $p_1 \equiv U \circ p_1'$. $RA5$ or $RA9$ rewriting rules in Table 3 can be applied. So p is not a normal form;
  - Subcase $p_1 \equiv \phi \cdot p_1'$. $RG1$, $RG3$, $RG4$, $RG5$, $RG7$, or $RG8-9$ rewriting rules can be applied. So p is not a normal form;
  - Subcase $p_1 \equiv p_1' + p_1''$. $RA4$, $RA6$, $RG2$, or $RG6$ rewriting rules in Table 3 can be applied. So p is not a normal form;
  - Subcase $p_1 \equiv p_1' \oplus p_1''$. $RP1$-$RP9$ rewrite rules in Table 7 can be applied. So p is not a normal form.
- Case $p' \equiv p_1 + p_2$. By induction on the structure of the basic APT terms both p_1 and p_2, all subcases will lead to that p' would be a basic APT term, which contradicts the assumption that p' is not a basic APT term.
- Case $p' \equiv p_1 \oplus p_2$. By induction on the structure of the basic APT terms both p_1 and p_2, all subcases will lead to that p' would be a basic APT term, which contradicts the assumption that p' is not a basic APT term.

□

4.3 Soundness and Completeness

Theorem 4.4 (Generalization of APT with respect to BAT)**.** *APT is a generalization of BAT.*

Proof. It follows from the following three facts.

1. The transition rules of BAT Table 4 are all source-dependent;
2. The sources of the transition rules for APT in Table 5 contain an occurrence of $\oplus$;
3. The transition rules of APT are all source-dependent.

So, APT is a generalization of BAT, i.e., BAT is an embedding of APT, as desired. □

Theorem 4.5 (Soundness of APT modulo pomset bisimulation equivalence)**.** *The axiomatization of APT is sound modulo pomset bisimulation equivalence, i.e., let x and y be APT terms. If $APT \vdash x = y$, then $x \underline{\leftrightarrow} y$.*

Proof. Since pomset bisimulations $\underline{\leftrightarrow}$ is both an equivalent and a congruent relation, we only need to check if each axiom in Table 6 is sound modulo pomset bisimulation equivalence $\underline{\leftrightarrow}$. The proof is trivial and we omit it. □

Theorem 4.6 (Completeness of APT modulo pomset bisimulation equivalence)**.** *The axiomatization of APT is complete modulo pomset bisimulation equivalence, i.e., let p and q be closed APT terms, if $p \underline{\leftrightarrow} q$ then $p = q$.*

Proof. Firstly, by the elimination theorem of APT (see Theorem 4.3), we know that for each closed APT term p, there exists a closed basic APT term p', such that $APT \vdash p = p'$, so, we only need to consider closed basic APT terms.

The basic terms (see Definition 4.2) modulo associativity and commutativity (AC) of conflict + (defined by axioms $A1$ and $A2$ in Table 2), and these equivalences is denoted by $=_{AC}$. Then, each equivalence class s modulo AC of + has the following normal form

$$s_1 + \cdots + s_k$$

with each s_i either an atomic event or of the form

$$t_1 \circ \cdots \circ t_m$$

with each t_j either an atomic event or of the form

$$u_1 \oplus \cdots \oplus u_\ell$$

with each u_ℓ an atomic event, and each s_i is called the summand of s.

Now, we prove that for normal forms n and n', if $n \underline{\leftrightarrow} n'$ then $n =_{AC} n'$. It is sufficient to induct on the sizes of n and n'.

- Consider a summand U of n. Then $\langle n, s\rangle \xrightarrow{U} \langle \surd, s'\rangle$, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \xrightarrow{U} \langle \surd, s\rangle$, meaning that n' also contains the summand U.
- Consider a summand ϕ of n. Then $\langle n, s\rangle \to \langle \surd, s\rangle$, if $test(\phi, s)$ holds, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \to \langle \surd, s\rangle$, if $test(\phi, s)$ holds, meaning that n' also contains the summand ϕ.
- Consider a summand $t_1 \cdot t_2$ of n,
  - if $t_1 \equiv \alpha'$, then $\langle n, s\rangle \xrightarrow{\alpha'} \langle t_2, s'\rangle$, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \xrightarrow{\alpha'} \langle t_2', s'\rangle$ with $t_2 \underline{\leftrightarrow} t_2'$, meaning that n' contains a summand $\alpha' \cdot t_2'$. Since t_2 and t_2' are normal forms and have sizes smaller than n and n', by the induction hypotheses if $t_2 \underline{\leftrightarrow} t_2'$ then $t_2 =_{AC} t_2'$;
  - if $t_1 \equiv \phi'$, then $\langle n, s\rangle \to \langle t_2, s\rangle$, if $test(\phi', s)$ holds, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \to \langle t_2', s\rangle$ with $t_2 \underline{\leftrightarrow} t_2'$, if $test(\phi', s)$ holds, meaning that n' contains a summand $\phi' \cdot t_2'$. Since t_2 and t_2' are normal forms and have sizes smaller than n and n', by the induction hypotheses if $t_2 \underline{\leftrightarrow} t_2'$ then $t_2 =_{AC} t_2'$;
  - if $t_1 \equiv \alpha_1 \oplus \cdots \oplus \alpha_{ell}$, then $\langle n, s\rangle \xrightarrow{\{\alpha_1, \cdots, \alpha_{ell}\}} \langle t_2, s'\rangle$, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \xrightarrow{\{\alpha_1, \cdots, \alpha_{ell}\}} \langle t_2', s'\rangle$ with $t_2 \underline{\leftrightarrow} t_2'$, meaning that n' contains a summand $(\alpha_1 \oplus \cdots \oplus \alpha_{ell}) \cdot t_2'$. Since t_2 and t_2' are normal forms and have sizes smaller than n and n', by the induction hypotheses if $t_2 \underline{\leftrightarrow} t_2'$ then $t_2 =_{AC} t_2'$;
  - if $t_1 \equiv \phi_1 \oplus \cdots \oplus \phi_{ell}$, then $\langle n, s\rangle \to \langle t_2, s\rangle$, if $test(\phi_1, s), \cdots, test(\phi_{ell}, s)$ hold, so $n \underline{\leftrightarrow} n'$ implies $\langle n', s\rangle \to \langle t_2', s\rangle$ with $t_2 \underline{\leftrightarrow} t_2'$, if $test(\phi_1, s), \cdots, test(\phi_{ell}, s)$ hold, meaning that n' contains a summand $(\phi_1 \oplus \cdots \oplus \phi_{ell}) \cdot t_2'$. Since t_2 and t_2' are normal forms and have sizes smaller than n and n', by the induction hypotheses if $t_2 \underline{\leftrightarrow} t_2'$ then $t_2 =_{AC} t_2'$.

So, we get $n =_{AC} n'$.

Finally, let s and t be basic APT terms, and $s \underline{\leftrightarrow} t$, there are normal forms n and n', such that $s = n$ and $t = n'$. The soundness theorem of parallelism modulo pomset bisimulation equivalence (see Theorem 4.5) yields $s \underline{\leftrightarrow} n$ and $t \underline{\leftrightarrow} n'$, so $n \underline{\leftrightarrow} s \underline{\leftrightarrow} t \underline{\leftrightarrow} n'$. Since if $n \underline{\leftrightarrow} n'$ then $n =_{AC} n'$, $s = n =_{AC} n' = t$, as desired. □

5 RECURSION

In this section, we introduce recursion to capture infinite computations based on APT.

In the following, E, F, G are recursion specifications, X, Y, Z are recursive variables.

Definition 5.1 (Recursive specification)**.** *A recursive specification is a finite set of recursive equations*

$$X_1 = t_1(X_1, \cdots, X_n)$$
$$\cdots$$
$$X_n = t_n(X_1, \cdots, X_n)$$

where the left-hand sides of X_i are called recursion variables, and the right-hand sides $t_i(X_1, \cdots, X_n)$ are process terms in APT with possible occurrences of the recursion variables $X_1, \cdots, X_n$.

Definition 5.2 (Solution)**.** *Processes $p_1, \cdots, p_n$ are solutions for a recursive specification $\{X_i = t_i(X_1, \cdots, X_n) | i \in \{1, \cdots, n\}\}$ (with respect to pomset bisimulation equivalence $\underline{\leftrightarrow}$) if $p_i \underline{\leftrightarrow} t_i(p_1, \cdots, p_n)$ for $i \in \{1, \cdots, n\}$.*

Definition 5.3 (Guarded recursive specification)**.** *A recursive specification*

$$X_1 = t_1(X_1, \cdots, X_n)$$
$$\cdots$$
$$X_n = t_n(X_1, \cdots, X_n)$$

is guarded if the right-hand sides of its recursive equations can be adapted to the form by applications of the axioms in APT and replacing recursion variables by the right-hand sides of their recursive equations, and there does not exist an infinite sequence of $\mathbf{I}$-transitions $\langle X|E\rangle \to \langle X'|E\rangle \to \langle X''|E\rangle \to \cdots$,

$$U_1 \circ s_1(X_1, \cdots, X_n) + \cdots + U_k \circ s_k(X_1, \cdots, X_n) + V_1 + \cdots + V_\ell$$

where $U_1, \cdots, U_k, V_1, \cdots, V_\ell \in \mathsf{Pom}$, and the sum above is allowed to be empty, in which case it represents the deadlock $\mathbf{0}$.

Definition 5.4 (Linear recursive specification)**.** *A recursive specification is linear if its recursive equations are of the form*

$$U_1 \circ X_1 + \cdots + U_k \circ X_k + V_1 + \cdots + V_\ell$$

where $U_1, \cdots, U_k, V_1, \cdots, V_\ell \in \mathsf{Pom}$, and the sum above is allowed to be empty, in which case it represents the deadlock $\mathbf{0}$.

For a guarded recursive specifications E with the form

$$X_1 = t_1(X_1, \cdots, X_n)$$
$$\cdots$$
$$X_n = t_n(X_1, \cdots, X_n)$$

the behavior of the solution $\langle X_i|E\rangle$ for the recursion variable X_i in E, where $i \in \{1, \cdots, n\}$, is exactly the behavior of their right-hand sides $t_i(X_1, \cdots, X_n)$, which is captured by the two transition rules in Table 8.

Theorem 5.5 (Conservativity of APT with guarded recursion)**.** *APT with guarded recursion is a conservative extension of APT.*

Proof. Since the transition rules of APT are source-dependent, and the transition rules for guarded recursion in Table 8 contain only a fresh constant in their source, so the transition rules of APT with guarded recursion are conservative extensions of those of APT. □

$$\frac{t_i(\langle X_1|E\rangle,\cdots,\langle X_n|E\rangle)\xrightarrow{U}\surd}{\langle X_i|E\rangle\xrightarrow{U}\surd}$$

$$\frac{t_i(\langle X_1|E\rangle,\cdots,\langle X_n|E\rangle)\xrightarrow{U}y}{\langle X_i|E\rangle\xrightarrow{U}y}$$

Table 8: Transition rules of guarded recursion

Theorem 5.6 (Congruence theorem of APT with guarded recursion)**.** *Pomset bisimulation equivalence $\underline{\leftrightarrow}$ is a congruences with respect to APT with guarded recursion.*

Proof. Since the PTSS of APT with guarded recursion in Table 8 is positive after reduction and in panth format, pomset bisimulation equivalence is a congruence with respect to APT with guarded recursion. □

The RDP (Recursive Definition Principle) and the RSP (Recursive Specification Principle) are shown in Table 9.

Table 9: Recursive definition and specification principle

No.	Axioms
RDP	$\langle X_i\|E\rangle = t_i(\langle X_1\|E\rangle,\cdots,\langle X_n\|E\rangle) \quad (i\in\{1,\cdots,n\})$
RSP	if $y_i = t_i(y_1,\cdots,y_n)$ for $i\in\{1,\cdots,n\}$, then $y_i = \langle X_i\|E\rangle \quad (i\in\{1,\cdots,n\})$

end of Recursive definition and specification principle

RDP follows immediately from the two transition rules for guarded recursion, which express that $\langle X_i|E\rangle$ and $t_i(\langle X_1|E\rangle,\cdots,\langle X_n|E\rangle)$ have the same initial transitions for $i\in\{1,\cdots,n\}$. RSP follows from the fact that guarded recursive specifications have only one solution.

Theorem 5.7 (Soundness of APT with guarded recursion)**.** *Let x and y be APT with guarded recursion terms. If APT with guarded recursion $\vdash x = y$, then $x\underline{\leftrightarrow}y$.*

Proof. Since pomset bisimulation equivalence is both an equivalent and a congruent relation with respect to APT with guarded recursion, we only need to check if each axiom in Table 9 is sound modulo pomset bisimulation equivalence. The proof is trivial and we omit it. □

Theorem 5.8 (Completeness of APT with linear recursion)**.** *Let p and q be closed APT with linear recursion terms, then, if $p\underline{\leftrightarrow}q$ then $p = q$.*

Proof. Firstly, we know that each process term in APT with linear recursion is equal to a process term $\langle X_1|E\rangle$ with E a linear recursive specification.

Let E_1 consist of recursive equations $X = t_X$ for $X\in\mathcal{X}$ and E_2 consists of recursion equations $Y = t_Y$ for $Y\in\mathcal{Y}$. Let the linear recursive specification E consist of recursion equations $Z_{XY} = t_{XY}$, and $\langle X|E_1\rangle\sim_s\langle Y|E_2\rangle$, and t_{XY} consists of the following summands:

1. t_{XY} contains a summand $(\alpha_1\oplus\cdots\oplus\alpha_m)Z_{X'Y'}$ iff t_X contains the summand $(\alpha_1\oplus\cdots\oplus\alpha_m)X'$ and t_Y contains the summand $(\alpha_1\oplus\cdots\oplus\alpha_m)Y'$ such that $\langle X'|E_1\rangle\sim_s\langle Y'|E_2\rangle$;
2. t_{XY} contains a summand $\beta_1\oplus\cdots\oplus\beta_n$ iff t_X contains the summand $\beta_1\oplus\cdots\oplus\beta_n$ and t_Y contains the summand $\beta_1\oplus\cdots\oplus\beta_n$.

$$\frac{}{\langle \tau_\phi, s\rangle \to \langle \surd, s\rangle} \text{ if } test(\tau_\phi, s)$$

$$\frac{}{\langle \tau, s\rangle \xrightarrow{\tau} \langle \surd, \tau(s)\rangle}$$

Table 10: Transition rule of the silent step

Let σ map recursion variable X in E_1 to $\langle X|E_1\rangle$, and let ψ map recursion variable Z_{XY} in E to $\langle X|E_1\rangle$. So, $\sigma((\alpha_1\oplus\cdots\oplus\alpha_m)X') \equiv (\alpha_1\oplus\cdots\oplus\alpha_m)\langle X'|E_1\rangle \equiv \psi((\alpha_1\oplus\cdots\oplus\alpha_m)Z_{X'Y'})$, so by RDP, we get $\langle X|E_1\rangle = \sigma(t_X) = \psi(t_{XY})$. Then by RSP, $\langle X|E_1\rangle = \langle Z_{XY}|E\rangle$, particularly, $\langle X_1|E_1\rangle = \langle Z_{X_1Y_1}|E\rangle$. Similarly, we can obtain $\langle Y_1|E_2\rangle = \langle Z_{X_1Y_1}|E\rangle$. Finally, $\langle X_1|E_1\rangle = \langle Z_{X_1Y_1}|E\rangle = \langle Y_1|E_2\rangle$, as desired. □

6 ABSTRACTION

To abstract away from the internal implementations of a transformer, and verify that the transformer exhibits the desired external behaviors, the silent step τ and abstraction operator τ_I are introduced, where $I \subseteq \mathsf{Act} \cup \mathsf{Grd}$ denotes the internal events or guards. The silent step τ represents the internal actions, and τ_ϕ for internal guards, when we consider the external behaviors of a process, τ steps can be removed, i.e., τ steps must keep silent. The transition rule of τ is shown in Table 10. In the following, let the atomic action a range over $\mathsf{Act}\cup\{\mathbf{0}\}\cup\{\mathbf{1}\}\cup\{\tau\}$, and ϕ range over $\mathsf{Grd}\cup\{\tau\}$. We use $\tau(s)$ to denote $effect(\tau, s)$, for the fact that τ only change the state of internal data environment, that is, for the external data environments, $s = \tau(s)$.

Definition 6.1 (Weak pomset transitions)**.** *Let $p, q \in \mathsf{Proc}$ and $U \in \mathsf{Pom}$, $p \overset{U}{\Longrightarrow} q$ is called a weak pomset transition from p to q, where we define $\overset{a}{\Longrightarrow} \triangleq \xrightarrow{\tau^*}\xrightarrow{a}\xrightarrow{\tau^*}$. And $\overset{U}{\Longrightarrow} \triangleq \xrightarrow{\tau^*}\xrightarrow{U}\xrightarrow{\tau^*}$, for every $a \in U$.*

Definition 6.2 (Branching pomset bisimulation)**.** *Assume a special termination predicate $\downarrow$, and let $\surd$ represent a state with $\surd\downarrow$. Let S be a set of states, $p, q \in \mathsf{Proc}$, $U \in \mathsf{Pom}$ and $s, s' \in S$. A branching pomset bisimulation is a relation $R \subseteq \langle \mathsf{Proc}, S\rangle \times \langle \mathsf{Proc}, S\rangle$, such that:*

1. *if $(\langle p, s\rangle, \langle q, s\rangle) \in R$, and $\langle p, s\rangle \xrightarrow{U} \langle p', s'\rangle$ then*
   - *either $U \equiv \tau^*$, and $(\langle P', s'\rangle, \langle q, s\rangle) \in R$ with $s' \in \tau(s)$;*
   - *or there is a sequence of (zero or more) τ-transitions $\langle q, s\rangle \xrightarrow{\tau^*} \langle q^0, s^0\rangle$, such that $(\langle p, s\rangle, \langle q^0, s^0\rangle) \in R$ and $\langle q^0, s^0\rangle \overset{U}{\Longrightarrow} \langle q', s'\rangle$ with $(\langle p', s'\rangle, \langle q', s'\rangle) \in R$;*
2. *if $(\langle p, s\rangle, \langle q, s\rangle) \in R$, and $\langle q, s\rangle \xrightarrow{U} \langle q', s'\rangle$ then*
   - *either $U \equiv \tau^*$, and $(\langle p, s\rangle, \langle q', s'\rangle) \in R$;*
   - *or there is a sequence of (zero or more) τ-transitions $\langle p, s\rangle \xrightarrow{\tau^*} \langle p^0, s^0\rangle$, such that $(\langle p^0, s^0\rangle, \langle q, s\rangle) \in R$ and $\langle p^0, s^0\rangle \overset{U}{\Longrightarrow} \langle p', s'\rangle$ with $(\langle p', s'\rangle, \langle q', s'\rangle) \in R$;*
3. *if $(\langle p, s\rangle, \langle q, s\rangle) \in R$ and $\langle p, s\rangle \downarrow$, then there is a sequence of (zero or more) τ-transitions $\langle q, s\rangle \xrightarrow{\tau^*} \langle q^0, s^0\rangle$ such that $(\langle p, s\rangle, \langle q^0, s^0\rangle) \in R$ and $\langle q^0, s^0\rangle \downarrow$;*
4. *if $(\langle p, s\rangle, \langle q, s\rangle) \in R$ and $\langle q, s\rangle \downarrow$, then there is a sequence of (zero or more) τ-transitions $\langle p, s\rangle \xrightarrow{\tau^*} \langle p^0, s^0\rangle$ such that $(\langle p^0, s^0\rangle, \langle C_2, s\rangle) \in R$ and $\langle p^0, s^0\rangle \downarrow$.*

We say that p, q are branching pomset bisimilar, written $p \underline{\leftrightarrow}_b$, if there exists a branching pomset bisimulation R, such that $(\langle p, s\rangle, \langle q, s\rangle) \in R$.

Definition 6.3 (Rooted branching pomset, step bisimulation)**.** *Assume a special termination predicate $\downarrow$, and let $\surd$ represent a state with $\surd\downarrow$. Let S be a set of states, $p,q \in \mathsf{Proc}$, $U \in \mathsf{Pom}$ and $s,s' \in S$. A rooted branching pomset bisimulation is a relation $R \subseteq \langle \mathsf{Proc}, S\rangle \times \langle \mathsf{Proc}, S\rangle$, such that:*

1. *if $(\langle p,s\rangle, \langle q,s\rangle) \in R$, and $\langle p,s\rangle \xrightarrow{U} \langle p',s'\rangle$ then $\langle q,s\rangle \xrightarrow{U} \langle q',s'\rangle$ with $\langle p',s'\rangle \underline{\leftrightarrow}_b \langle q',s'\rangle$;*
2. *if $(\langle p,s\rangle, \langle q,s\rangle) \in R$, and $\langle q,s\rangle \xrightarrow{U} \langle q',s'\rangle$ then $\langle p,s\rangle \xrightarrow{U} \langle p',s'\rangle$ with $\langle p',s'\rangle \underline{\leftrightarrow}_b \langle q',s'\rangle$;*
3. *if $(\langle p,s\rangle, \langle q,s\rangle) \in R$ and $\langle p,s\rangle\downarrow$, then $\langle q,s\rangle\downarrow$;*
4. *if $(\langle p,s\rangle, \langle q,s\rangle) \in R$ and $\langle q,s\rangle\downarrow$, then $\langle p,s\rangle\downarrow$.*

We say that p,q are rooted branching pomset bisimilar, written $p \underline{\leftrightarrow}_{rb} q$, if there exists a rooted branching pomset bisimulation R, such that $(\langle p,s\rangle, \langle q,s\rangle) \in R$.

The silent step τ as an atomic event, is introduced into E. Considering the recursive specification $X = \tau X$, τs, $\tau\tau s$, and $\tau\cdots s$ are all its solutions, that is, the solutions make the existence of τ-loops which cause unfairness. To prevent τ-loops, we extend the definition of linear recursive specification (Definition 5.4) to the guarded one.

Definition 6.4 (Guarded linear recursive specification)**.** *A recursive specification is linear if its recursive equations are of the form*

$$U_1 \circ X_1 + \cdots + U_k \circ X_k + V_1 + \cdots + V_\ell$$

where $U_1,\cdots,U_k,V_1,\cdots,V_\ell \in \mathsf{Pom}$, and the sum above is allowed to be empty, in which case it represents the deadlock $\mathbf{0}$.

A linear recursive specification E is guarded if there does not exist an infinite sequence of $\mathbf{I}$-transitions $\langle X|E\rangle \to \langle X'|E\rangle \to \langle X''|E\rangle \to \cdots$ and τ-transitions $\langle X|E\rangle \xrightarrow{\tau} \langle X'|E\rangle \xrightarrow{\tau} \langle X''|E\rangle \xrightarrow{\tau} \cdots$.

Theorem 6.5 (Conservativity of APT with silent step and guarded linear recursion)**.** *APT with silent step and guarded linear recursion is a conservative extension of APT with linear recursion.*

Proof. Since the transition rules of APT with linear recursion are source-dependent, and the transition rules for silent step in Table 10 contain only a fresh constant τ in their source, so the transition rules of APT with silent step and guarded linear recursion is a conservative extension of those of APT with linear recursion. □

Theorem 6.6 (Congruence theorem of APT with silent step and guarded linear recursion)**.** *Rooted branching pomset bisimulation equivalence $\underline{\leftrightarrow}_{rb}$ is a congruence with respect to APT with silent step and guarded linear recursion.*

Proof. Since the PTSS of APT with silent step and guarded linear recursion in Table 10 is positive after reduction and in RBB cool format, rooted branching pomset bisimulation equivalence $\underline{\leftrightarrow}_{rb}$ is a congruence with respect to APT with with silent step and guarded linear recursion. □

We design the axioms for the silent step τ in Table 11.

Table 11: Axioms of silent step

No.	Axioms
$B1$	$U \circ \tau = U$
$B2$	$U \circ (\tau \circ (x+y) + x) = U \circ (x+y)$
$B3$	$x \oplus \tau = x$
$B4$	$\tau \oplus x = x$
$G26$	$\tau_\phi \cdot x = x$
$G19$	$x \cdot \tau_\phi = x$

to be continued...

continuing: Axioms of silent step

No.	Axioms
$G20$	$x \parallel \tau_\phi = x$
$G21$	$\tau_\phi \parallel x = x$

end of Axioms of silent step

Theorem 6.7 (Soundness of APT with silent step and guarded linear recursion)**.** *Let x and y be APT with silent step and guarded linear recursion terms. If APT with silent step and guarded linear recursion $\vdash x = y$, then $x \underline{\leftrightarrow}_{rb} y$.*

Proof. Since rooted branching pomset bisimulation $\underline{\leftrightarrow}_{rb}$ is both an equivalent and a congruent relation with respect to APT with silent step and guarded linear recursion, we only need to check if each axiom in Table 11 is sound modulo rooted branching pomset bisimulation equivalence. The proof is trivial and we omit it. □

Theorem 6.8 (Completeness of APT with silent step and guarded linear recursion)**.** *Let p and q be closed APT with silent step and guarded linear recursion terms, then, if $p \underline{\leftrightarrow}_{rb} q$ then $p = q$.*

Proof. Firstly, we know that each process term in APT with silent step and guarded linear recursion is equal to a process term $\langle X_1|E\rangle$ with E a guarded linear recursive specification.

The recursive equation $W = \tau + \cdots + \tau$ with $W \notin X_1$ in E_1 and E_2, can be removed, and the corresponding summands aW are replaced by α, to get E_1' and E_2', by use of the axioms RDP, $A3$ and $B1$, and $\langle X|E_1\rangle = \langle X|E_1'\rangle$, $\langle Y|E_2\rangle = \langle Y|E_2'\rangle$.

Let E_1 consists of recursive equations $X = t_X$ for $X \in \mathcal{X}$ and E_2 consists of recursion equations $Y = t_Y$ for $Y \in \mathcal{Y}$, and are not the form $\tau + \cdots + \tau$. Let the guarded linear recursive specification E consists of recursion equations $Z_{XY} = t_{XY}$, and $\langle X|E_1\rangle \approx_{rbs} \langle Y|E_2\rangle$, and t_{XY} consists of the following summands:

1. t_{XY} contains a summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)Z_{X'Y'}$ iff t_X contains the summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)X'$ and t_Y contains the summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)Y'$ such that $\langle X'|E_1\rangle \approx_{rbs} \langle Y'|E_2\rangle$;
2. t_{XY} contains a summand $\beta_1 \oplus \cdots \oplus \beta_n$ iff t_X contains the summand $\beta_1 \oplus \cdots \oplus \beta_n$ and t_Y contains the summand $\beta_1 \oplus \cdots \oplus \beta_n$;
3. t_{XY} contains a summand $\tau Z_{X'Y}$ iff $XY \notin X_1Y_1$, t_X contains the summand $\tau X'$, and $\langle X'|E_1\rangle \approx_{rbs} \langle Y|E_2\rangle$;
4. t_{XY} contains a summand $\tau Z_{XY'}$ iff $XY \notin X_1Y_1$, t_Y contains the summand $\tau Y'$, and $\langle X|E_1\rangle \approx_{rbs} \langle Y'|E_2\rangle$.

Since E_1 and E_2 are guarded, E is guarded. Constructing the process term u_{XY} consist of the following summands:

1. u_{XY} contains a summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)\langle X'|E_1\rangle$ iff t_X contains the summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)X'$ and t_Y contains the summand $(\alpha_1 \oplus \cdots \oplus \alpha_m)Y'$ such that $\langle X'|E_1\rangle \approx_{rbs} \langle Y'|E_2\rangle$;
2. u_{XY} contains a summand $\beta_1 \oplus \cdots \oplus \beta_n$ iff t_X contains the summand $\beta_1 \oplus \cdots \oplus \beta_n$ and t_Y contains the summand $\beta_1 \oplus \cdots \oplus \beta_n$;
3. u_{XY} contains a summand $\tau\langle X'|E_1\rangle$ iff $XY \notin X_1Y_1$, t_X contains the summand $\tau X'$, and $\langle X'|E_1\rangle \approx_{rbs} \langle Y|E_2\rangle$.

Let the process term s_{XY} be defined as follows:

1. $s_{XY} \triangleq \tau\langle X|E_1\rangle + u_{XY}$ iff $XY \notin X_1Y_1$, t_Y contains the summand $\tau Y'$, and $\langle X|E_1\rangle \approx_{rbs} \langle Y'|E_2\rangle$;

$$\frac{\langle x,s\rangle \xrightarrow{\alpha} \langle \surd, s'\rangle}{\langle \tau_I(x),s\rangle \xrightarrow{\alpha} \langle \surd, s'\rangle} \quad \alpha \notin I \qquad \frac{\langle x,s\rangle \xrightarrow{\alpha} \langle x', s'\rangle}{\langle \tau_I(x),s\rangle \xrightarrow{\alpha} \langle \tau_I(x'), s'\rangle} \quad \alpha \notin I$$

$$\frac{\langle x,s\rangle \xrightarrow{\alpha} \langle \surd, s'\rangle}{\langle \tau_I(x),s\rangle \xrightarrow{\tau} \langle \surd, \tau(s)\rangle} \quad \alpha \in I \qquad \frac{\langle x,s\rangle \xrightarrow{\alpha} \langle x', s'\rangle}{\langle \tau_I(x),s\rangle \xrightarrow{\tau} \langle \tau_I(x'), \tau(s)\rangle} \quad \alpha \in I$$

Table 12: Transition rule of the abstraction operator

2. $s_{XY} \triangleq \langle X|E_1\rangle$, otherwise.

So, $\langle X|E_1\rangle = \langle X|E_1\rangle + u_{XY}$, and $(\alpha_1 \oplus \cdots \oplus \alpha_m)(\tau\langle X|E_1\rangle + u_{XY}) = (\alpha_1 \oplus \cdots \oplus \alpha_m)((\tau\langle X|E_1\rangle + u_{XY}) + u_{XY}) = (\alpha_1 \oplus \cdots \oplus \alpha_m)(\langle X|E_1\rangle + u_{XY}) = (\alpha_1 \oplus \cdots \oplus \alpha_m)\langle X|E_1\rangle$, hence, $(\alpha_1 \oplus \cdots \oplus \alpha_m)s_{XY} = (\alpha_1 \oplus \cdots \oplus \alpha_m)\langle X|E_1\rangle$.

Let σ map recursion variable X in E_1 to $\langle X|E_1\rangle$, and let ψ map recursion variable Z_{XY} in E to s_{XY}. It is sufficient to prove $s_{XY} = \psi(t_{XY})$ for recursion variables Z_{XY} in E. Either $XY \equiv X_1Y_1$ or $XY \not\equiv X_1Y_1$, we all can get $s_{XY} = \psi(t_{XY})$. So, $s_{XY} = \langle Z_{XY}|E\rangle$ for recursive variables Z_{XY} in E is a solution for E. Then by RSP, particularly, $\langle X_1|E_1\rangle = \langle Z_{X_1Y_1}|E\rangle$. Similarly, we can obtain $\langle Y_1|E_2\rangle = \langle Z_{X_1Y_1}|E\rangle$. Finally, $\langle X_1|E_1\rangle = \langle Z_{X_1Y_1}|E\rangle = \langle Y_1|E_2\rangle$, as desired. □

The unary abstraction operator τ_I ($I \subseteq \mathsf{Act}$) renames all atomic events in I into τ. APT with silent step and abstraction operator is called APT_τ. The transition rules of operator τ_I are shown in Table 12.

Theorem 6.9 (Conservativity of APT_τ with guarded linear recursion)**.** *APT_τ with guarded linear recursion is a conservative extension of APT with silent step and guarded linear recursion.*

Proof. Since the transition rules of APT with silent step and guarded linear recursion are source-dependent, and the transition rules for abstraction operator in Table 12 contain only a fresh operator τ_I in their source, so the transition rules of APT_τ with guarded linear recursion is a conservative extension of those of APT with silent step and guarded linear recursion. □

Theorem 6.10 (Congruence theorem of APT_τ with guarded linear recursion)**.** *Rooted branching pomset bisimulation equivalence $\underline{\leftrightarrow}_{rb}$ is a congruence with respect to APT_τ with guarded linear recursion.*

Proof. Since the PTSS of APT_τ with guarded linear recursion in Table 12 is positive after reduction and in RBB cool format, rooted branching pomset bisimulation equivalence $\underline{\leftrightarrow}_{rb}$ is a congruence with respect to APT_τ with guarded linear recursion. □

We design the axioms for the abstraction operator τ_I in Table 13.

Table 13: Axioms of abstraction

No.	Axioms
$TI1$	$\alpha \notin I \quad \tau_I(\alpha) = \alpha$
$TI2$	$\alpha \in I \quad \tau_I(\alpha) = \tau$
$TI3$	$\tau_I(\mathbf{0}) = \mathbf{0}$
$TI4$	$\tau_I(\mathbf{1}) = \mathbf{1}$
$TI5$	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$
$TI6$	$\tau_I(x \circ y) = \tau_I(x) \circ \tau_I(y)$
$TI7$	$\tau_I(x \oplus y) = \tau_I(x) \oplus \tau_I(y)$
$G22$	$\phi \notin I \quad \tau_I(\phi) = \phi$
$G23$	$\phi \in I \quad \tau_I(\phi) = \tau_\phi$

end of Axioms of abstraction

Theorem 6.11 (Soundness of APT_τ with guarded linear recursion)**.** *Let x and y be APT_τ with guarded linear recursion terms. If APT_τ with guarded linear recursion $\vdash x = y$, then $x \underline{\leftrightarrow}_{rb} y$.*

Proof. Since rooted branching pomset bisimulation $\underline{\leftrightarrow}_{rb}$ is both an equivalent and a congruent relation with respect to APT_τ with guarded linear recursion, we only need to check if each axiom in Table 13 is sound modulo rooted branching pomset bisimulation equivalence. The proof is trivial and we omit it. □

Though τ-loops are prohibited in guarded linear recursive specifications (see Definition 6.4) in a specifiable way, they can be constructed using the abstraction operator, for example, there exist τ-loops in the process term $\tau_{\{\alpha\}}(\langle X|X = aX\rangle)$. To avoid τ-loops caused by τ_I and ensure fairness, the concept of cluster and $CFAR$ (Cluster Fair Abstraction Rule) Vaandrager (1986) are still valid, we introduce them below.

Definition 6.12 (Cluster)**.** *Let E be a guarded linear recursive specification, and $I \subseteq \mathsf{Act}$. Two recursion variable X and Y in E are in the same cluster for I iff there exist sequences of transitions $\langle X|E\rangle \xrightarrow{\{\!|\beta_{11},\cdots,\beta_{1i}|\!\}} \cdots \xrightarrow{\{\!|\beta_{m1},\cdots,\beta_{mi}|\!\}} \langle Y|E\rangle$ and $\langle Y|E\rangle \xrightarrow{\{\!|\gamma_{11},\cdots,\gamma_{1j}|\!\}} \cdots \xrightarrow{\{\!|\gamma_{n1},\cdots,\gamma_{nj}|\!\}} \langle X|E\rangle$, where $\beta_{11},\cdots,\beta_{mi},\gamma_{11},\cdots,\gamma_{nj} \in I \cup \{\tau\}$.*

$\alpha_1 \oplus \cdots \oplus \alpha_k$ or $(\alpha_1 \oplus \cdots \oplus \alpha_k)X$ is an exit for the cluster C iff: (1) $\alpha_1 \oplus \cdots \oplus \alpha_k$ or $(\alpha_1 \oplus \cdots \oplus \alpha_k)X$ is a summand at the right-hand side of the recursive equation for a recursion variable in C, and (2) in the case of $(\alpha_1 \oplus \cdots \oplus \alpha_k)X$, either $\alpha_\ell \notin I \cup \{\tau\} (\ell \in \{1, 2, \cdots, k\})$ or $X \notin C$.

Table 14: Cluster fair abstraction rule

No.	Axioms
$CFAR$	If X is in a cluster for I with exits $\{(\alpha_{11} \oplus \cdots \oplus \alpha_{1i})Y_1, \cdots, (\alpha_{m1} \oplus \cdots \oplus \alpha_{mi})Y_m, \beta_{11} \oplus \cdots \oplus \beta_{1j}, \cdots, \beta_{n1} \oplus \cdots \oplus \beta_{nj}\}$, then $\tau \circ \tau_I(\langle X|E\rangle) =$ $\tau \circ \tau_I((\alpha_{11} \oplus \cdots \oplus \alpha_{1i})\langle Y_1|E\rangle + \cdots + (\alpha_{m1} \oplus \cdots \oplus \alpha_{mi})\langle Y_m|E\rangle + \beta_{11} \oplus \cdots \oplus \beta_{1j} + \cdots + \beta_{n1} \oplus \cdots \oplus \beta_{nj})$

end of Cluster fair abstraction rule

Theorem 6.13 (Soundness of $CFAR$)**.** *$CFAR$ is sound modulo rooted branching pomset bisimulation equivalence $\underline{\leftrightarrow}_{rb}$.*

Proof. Let X be in a cluster for I with exits $\{(\alpha_{11} \oplus \cdots \oplus \alpha_{1i})Y_1, \cdots, (\alpha_{m1} \oplus \cdots \oplus \alpha_{mi})Y_m, \beta_{11} \oplus \cdots \oplus \beta_{1j}, \cdots, \beta_{n1} \oplus \cdots \oplus \beta_{nj}\}$. Then $\langle X|E\rangle$ can execute a string of atomic events from $I \cup \{\tau\}$ inside the cluster of X, followed by an exit $(\alpha_{i'1} \oplus \cdots \oplus \alpha_{i'i})Y_{i'}$ for $i' \in \{1, \cdots, m\}$ or $\beta_{j'1} \oplus \cdots \oplus \beta_{j'j}$ for $j' \in \{1, \cdots, n\}$. Hence, $\tau_I(\langle X|E\rangle)$ can execute a string of τ^* inside the cluster of X, followed by an exit $\tau_I((\alpha_{i'1} \oplus \cdots \oplus \alpha_{i'i})\langle Y_{i'}|E\rangle)$ for $i' \in \{1, \cdots, m\}$ or $\tau_I(\beta_{j'1} \oplus \cdots \oplus \beta_{j'j})$ for $j' \in \{1, \cdots, n\}$. And these τ^* are non-initial in $\tau\tau_I(\langle X|E\rangle)$, so they are truly silent by the axiom $B1$, we obtain $\tau\tau_I(\langle X|E\rangle) \approx_{rbs} \tau \circ \tau_I((\alpha_{11} \oplus \cdots \oplus \alpha_{1i})\langle Y_1|E\rangle + \cdots + (\alpha_{m1} \oplus \cdots \oplus \alpha_{mi})\langle Y_m|E\rangle + \beta_{11} \oplus \cdots \oplus \beta_{1j} + \cdots + \beta_{n1} \oplus \cdots \oplus \beta_{nj})$, as desired. □

Theorem 6.14 (Completeness of APT_τ with guarded linear recursion and $CFAR$)**.** *Let p and q be closed APT_τ with guarded linear recursion and $CFAR$ terms, then, if $p \underline{\leftrightarrow}_{rb} q$ then $p = q$.*

Proof. Firstly, in the proof the Theorem 6.8, we know that each process term p in APT with silent step and guarded linear recursion is equal to a process term $\langle X_1|E\rangle$ with E a guarded linear recursive specification. And we prove if $\langle X_1|E_1\rangle \approx_{rbs} \langle Y_1|E_2\rangle$, then $\langle X_1|E_1\rangle = \langle Y_1|E_2\rangle$

The only new case is $p \equiv \tau_I(q)$. Let $q = \langle X|E\rangle$ with E a guarded linear recursive specification, so $p = \tau_I(\langle X|E\rangle)$. Then the collection of recursive variables in E can be divided into its clusters $C_1, \cdots, C_N$ for I. Let

$(\alpha_{1i1} \oplus \cdots \oplus \alpha_{k_{i1}i1})Y_{i1} + \cdots + (\alpha_{1im_i} \oplus \cdots \oplus \alpha_{k_{im_i}im_i})Y_{im_i} + \beta_{1i1} \oplus \cdots \oplus \beta_{\ell_{i1}i1} + \cdots + \beta_{1im_i} \oplus \cdots \oplus \beta_{\ell_{im_i}im_i}$

be the conflict composition of exits for the cluster C_i, with $i \in \{1, \cdots, N\}$.

For $Z \in C_i$ with $i \in \{1, \cdots, N\}$, we define

$s_Z \triangleq (\alpha_{\hat{1}i1} \oplus \cdots \oplus \alpha_{\hat{k_{i1}}i1})\tau_I(\langle Y_{i1}|E\rangle) + \cdots + (\alpha_{1\hat{im_i}} \oplus \cdots \oplus \alpha_{k_{\hat{im_i}}im_i})\tau_I(\langle Y_{im_i}|E\rangle) + \beta_{\hat{1i1}} \oplus \cdots \oplus \beta_{\hat{\ell_{i1}}i1} + \cdots + \beta_{1\hat{im_i}} \oplus \cdots \oplus \beta_{\ell_{\hat{im_i}}im_i}$

For $Z \in C_i$ and $\alpha_1, \cdots, \alpha_j \in \mathsf{Act} \cup \{\tau\}$ with $j \in \mathbb{N}$, we have

$(\alpha_1 \oplus \cdots \oplus \alpha_j)\tau_I(\langle Z|E\rangle)$

$= (\alpha_1 \oplus \cdots \oplus \alpha_j)\tau_I((\alpha_{1i1} \oplus \cdots \oplus \alpha_{k_{i1}i1})\langle Y_{i1}|E\rangle + \cdots + (\alpha_{1im_i} \oplus \cdots \oplus \alpha_{k_{im_i}im_i})\langle Y_{im_i}|E\rangle + \beta_{1i1} \oplus \cdots \oplus \beta_{\ell_{i1}i1} + \cdots + \beta_{1im_i} \oplus \cdots \oplus \beta_{\ell_{im_i}im_i})$

$= (\alpha_1 \oplus \cdots \oplus \alpha_j)s_Z$

Let the linear recursive specification F contain the same recursive variables as E, for $Z \in C_i$, F contains the following recursive equation

$Z = (\alpha_{\hat{1}i1} \oplus \cdots \oplus \alpha_{\hat{k_{i1}}i1})Y_{i1} + \cdots + (\alpha_{1\hat{im_i}} \oplus \cdots \oplus \alpha_{k_{\hat{im_i}}im_i})Y_{im_i} + \beta_{\hat{1i1}} \oplus \cdots \oplus \beta_{\hat{\ell_{i1}}i1} + \cdots + \beta_{1\hat{im_i}} \oplus \cdots \oplus \beta_{\ell_{\hat{im_i}}im_i}$

It is easy to see that there is no sequence of one or more τ-transitions from $\langle Z|F\rangle$ to itself, so F is guarded.

For

$s_Z = (\alpha_{\hat{1}i1} \oplus \cdots \oplus \alpha_{\hat{k_{i1}}i1})Y_{i1} + \cdots + (\alpha_{1\hat{im_i}} \oplus \cdots \oplus \alpha_{k_{\hat{im_i}}im_i})Y_{im_i} + \beta_{\hat{1i1}} \oplus \cdots \oplus \beta_{\hat{\ell_{i1}}i1} + \cdots + \beta_{1\hat{im_i}} \oplus \cdots \oplus \beta_{\ell_{\hat{im_i}}im_i}$

is a solution for F. So, $(\alpha_1 \oplus \cdots \oplus \alpha_j)\tau_I(\langle Z|E\rangle) = (\alpha_1 \oplus \cdots \oplus \alpha_j)s_Z = (\alpha_1 \oplus \cdots \oplus \alpha_j)\langle Z|F\rangle$.

So,

$\langle Z|F\rangle = (\alpha_{\hat{1}i1} \oplus \cdots \oplus \alpha_{\hat{k_{i1}}i1})\langle Y_{i1}|F\rangle + \cdots + (\alpha_{1\hat{im_i}} \oplus \cdots \oplus \alpha_{k_{\hat{im_i}}im_i})\langle Y_{im_i}|F\rangle + \beta_{\hat{1i1}} \oplus \cdots \oplus \beta_{\hat{\ell_{i1}}i1} + \cdots + \beta_{1\hat{im_i}} \oplus \cdots \oplus \beta_{\ell_{\hat{im_i}}im_i}$

Hence, $\tau_I(\langle X|E\rangle = \langle Z|F\rangle)$, as desired. □

REFERENCES

D. Hsu C. Sanford and M. Telgarsky. Transformers, parallel computation, and logarithmic depth. *Under review*, 2024. Preprint, available at arXiv:2402.09268.

D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. *Computational Problems in Abstract Algebra*, pp. 263–297, 1970.

W. Merrill and A. Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11.

J. Pérez, P. Barceló, and J. Marinkovic. Attention is turing complete. *Journal of Machine Learning Research*, 22:1–35, 2021.

F.W. Vaandrager. Verification of two communication protocols by means of process algebra. Technical Report Report CS-R8608, CWI, Amsterdam, 1986.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

A. Yang, C. Watson, A. Xue, S. Bhattamishra, J. Llarena, W. Merrill, E. D. S. Ferreira, A. S., and D. Chiang. The transformer cookbook, 2025. URL https://arxiv.org/abs/2510.00368.