PhD Viva

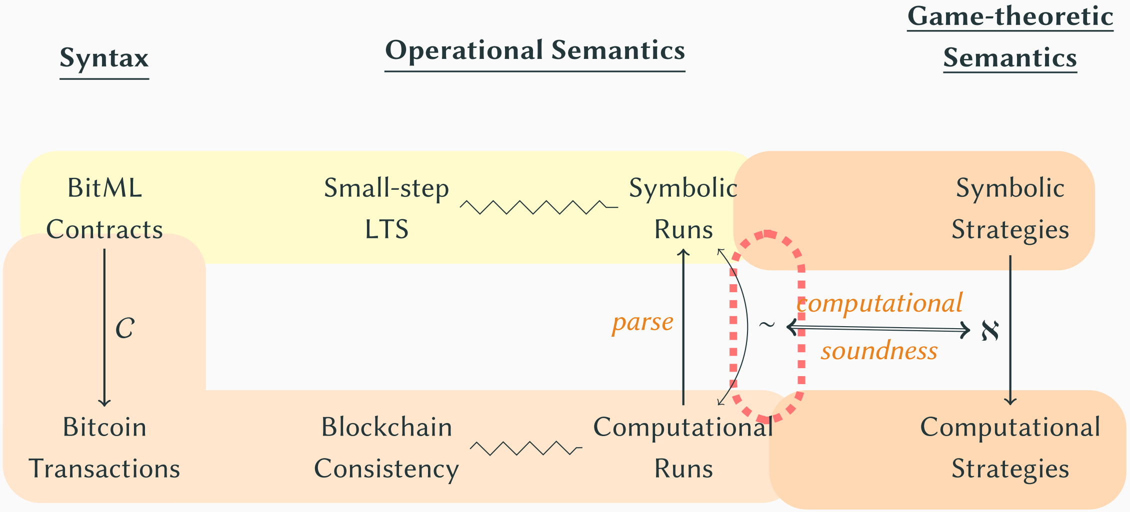
Orestis Melkonian

December 8, 2023

THE UNIVERSITY of EDINBURGH

INPUT | OUTPUT

BitML Progress



BitML paper: coherence relation

Definition 20 (Coherence). We inductively define the relation $coher(R^s, R^c, r, txout, sechash, \kappa)$, where (i) R^s is a symbolic run, (ii) R^c is a computational run, (iii) r is a randomness source, (iv) $txout$ is an injective function from names x (occurring in R^s) to transaction ouputs (T, o) (where T occurs in R^c), respecting values; (v) $sechash$ is a mapping from secret names a (occurring in R^s) to bitstrings; (vi) κ maps triples $(\{G\}C, D, \mathsf{A})$, where D is a subterm of C, to public keys.

Base case: $coher(R^s, R^c, r, txout, sechash, \kappa)$ holds if all the following conditions hold: (i) $R^s = \Gamma_0 \mid 0$, with Γ_0 initial; (ii) $R^c = \mathsf{T}_0 \cdots$ initial; (iii) all the public keys in R^c are generated from r, according to Definition 13; (iv) $txout$ maps exactly the x of $\langle \mathsf{A}, v\rangle_x$ in Γ_0 to an output in T_0 of value v₿, and spendable with $\hat{K}_{\mathsf{A}}(r_{\mathsf{A}})$; (v) dom $sechash = \emptyset$; (vi) dom $\kappa = \emptyset$.

Inductive case: $coher(\dot{R}^s \xrightarrow{\alpha} \Gamma \mid t, \dot{R}^c \lambda^c, r, txout, sechash, \kappa)$ holds if $coher(\dot{R}^s, \dot{R}^c, r, txout', sechash', \kappa')$ and one of the following cases applies.

(1) $\alpha = advertise(\{G\}C)$, $\lambda^c = \mathsf{A} \to * : C$, where C is obtained by encoding $\{G\}C$ as a bitstring, representing each x in it as the transaction output $txout'(x)$. Further, $txout' = txout$, $sechash' = sechash$, and $\kappa' = \kappa$.

(2) $\alpha = \mathsf{A} : \{G\}C, \Delta$, where: (i) for some B, $\dot{R}^c$ contains $\mathsf{B} \to * : C$, where C is obtained from $\{G\}C$ and $txout'$ as in Item 1. Note that $\dot{R}^c$ might contain several such messages; below, we let C represent the first occurrence. (ii) for some B, $\lambda^c = \mathsf{B} \to * : (C, \vec{h}, \vec{k})$ (signed by A), where $\vec{h}$ is a sequence comprising a bitstring h_i with $|h_i| = \eta$ for each secret a_i in Δ, and $\vec{k}$ is a sequence of keys, as the one produced by the stipulation protocol. We require that λ^c is the first occurrence, in the run $\dot{R}^c$, of such a message after C. (iii) Let N_i be the length of a_i fixed in Δ. If $N_i \neq \bot$, we require that $\dot{R}^c$ contains, for some B, a query to the oracle $\mathsf{B} \to \mathsf{O} : m_i$, and a subsequent reply $\mathsf{O} \to \mathsf{B} : h_i$ such that $|m_i| = \eta + N_i$. Otherwise, if $N_i = \bot$, we require that h_i does not occur as a reply from O to any query of length $\geq \eta$. (iv) No hash is reused: the h_i are pairwise distinct, and also distinct from $sechash'(b)$ for any $b \in$ dom $(sechash')$. (v) $txout = txout'$. (vi) $sechash$ extends $sechash'$ so that for each secret a_i we have $sechash(a_i) = h_i$. (vii) If $\mathsf{A} \in \mathsf{Hon}$, we define κ by extending κ' according to $\vec{k}$, so to record the public keys of all participants occurring in G for each subterm D of C. If κ' already defines such keys, or $\mathsf{A} \notin \mathsf{Hon}$, we let $\kappa = \kappa'$.

(3) $\alpha = \mathsf{A} : \{G\}C, x$, where: (i) $\lambda^c = \mathsf{B} \to * : m$ for some B, where m is the signature of the transaction T_{init} of ₿$_{\mathsf{adv}}(\{G\}C)$ relatively to the input x with $\hat{K}_{\mathsf{A}}(r_{\mathsf{A}})$. The parameters of the compiler are set as follows: part, $\mathsf{Part}\,G$ and in ₿$_{\mathsf{adv}}(\{G\}C)$. The needed compiler parameters are obtained as in Item 3. (iii) $sechash = sechash'$, $\kappa = \kappa'$, and $txout$ extends $txout'$, mapping z to T_{init}.

(5) $\alpha = \mathsf{A} : x, D$, where: (i) $\dot{R}^s$ contains $\langle C', v\rangle_x$ with $C' = D + \sum_i D_i$, for some $D = \mathsf{A} : D'$. (ii) In $\dot{R}^s$, we find that $\langle C', v\rangle_x$ has $\{G\}C$ as its ancestor advertisement. (iii) $\lambda^c = \mathsf{B} \to * : m$, where m is a signature with key $\kappa'(\{G\}C, D, \mathsf{A})$ of the first transaction T in ₿$_{\mathsf{D}}(D, D, \mathsf{T}', o, v, \mathsf{PartG}, 0)$, where $(\mathsf{T}', o) = txout'(x)$. The compiler parameters are obtained as in Item 3. (iv) $txout = txout'$, $sechash = sechash'$, and $\kappa = \kappa'$. (v) $\dot{R}^c$ contains $\mathsf{B} \to * : \mathsf{T}$ for some B, and m is the first signature of T in $\dot{R}^c \lambda^c$ after the first broadcast of T.

(6) $\alpha = put(\vec{x}, \vec{a}, y)$, where: (i) $\vec{x} = x_1 \cdots x_k$. (ii) In $\Gamma_{\dot{R}^s}$, the action α consumes $\langle D + C, v\rangle_y$ and the deposits $\langle \mathsf{A}_i, v_i\rangle_{x_i}$ to produce $\langle C', v'\rangle_{y'}$, where $D = \cdots : \mathtt{put} \cdots \mathtt{reveal} \cdots . C'$. Let t be maximum deadline in an $\mathtt{after}$ in front of D. (iii) In $\dot{R}^s$, we find that $\langle D + C, v\rangle_y$ has $\{G\}C''$ as its ancestor advertisement, for some G and C''. (iv) $\lambda^c = \mathsf{T}$ where T is the first transaction of ₿$_{\mathsf{C}}(C', D, \mathsf{T}', o, v', \vec{x}, \mathsf{PartG}, t)$, where $(\mathsf{T}', o) = txout'(y)$. The compiler parameters are obtained as in Item 3. (v) $txout$ extends $txout'$ so that y' is mapped to $(\mathsf{T}, 0)$, $sechash = sechash'$, and $\kappa = \kappa'$.

(7) $\alpha = \mathsf{A} : a$, where: (i) $\lambda^c = \mathsf{B} \to * : m$ from some B with $|m| \geq \eta$. (ii) $\dot{R}^c = \cdots (\mathsf{B} \to \mathsf{O} : m)(\mathsf{O} \to \mathsf{B} : sechash'(a)) \cdots$, for some B. (iii) $txout = txout'$, $sechash = sechash'$ and $\kappa = \kappa'$. (iv) In $\dot{R}^s$ we find an $\mathsf{A} : \{G\}C, \Delta$ action, with a in G, with a corresponding broadcast in $\dot{R}^c$ of $m' = (C, \vec{h}, \vec{k})$. (v) λ^c is the first broadcast of m in $\dot{R}^c$ after the first broadcast of m'.

(8) $\alpha = split(y)$, where: (i) In $\dot{R}^s$, the action α consumes $\langle D + C, v\rangle_y$ to obtain $\langle C_0, v_0\rangle_{x_0} \mid \cdots \mid \langle C_k, v_k\rangle_{x_k}$ where $D = \cdots : \mathtt{split}\ \vec{v} \to \vec{C}$ and $\vec{C} = C_0 \ldots C_k$. Let t be the maximum deadline in an $\mathtt{after}$ in front of D. (ii) In $\dot{R}^s$, we find that $\langle D + C, v\rangle_y$ has $\{G\}C'$ as its ancestor advertisement. (iii) $\lambda^c = \mathsf{T}$ where T is the first transaction of ₿$_{\mathsf{par}}(\vec{C}, D, \mathsf{T}', o, \mathsf{PartG}, t)$ where $(\mathsf{T}', o) = txout'(y)$. The compiler parameters are obtained as for Item 3. (iv) $txout$ extends $txout'$ mapping each x_i to (T, i), $sechash = sechash'$, and $\kappa = \kappa'$.

(9) $\alpha = withdraw(\mathsf{A}, v, y)$, where: (i) In $\dot{R}^s$, the action α consumes $\langle D + C, v\rangle_y$ to obtain $\langle \mathsf{A}, v\rangle_x$, where $D = \cdots : \mathtt{withdraw}\ \mathsf{A}$. (ii) In $\dot{R}^s$, we find that $\langle D + C, v\rangle_y$ has $\{G\}C'$ as its ancestor advertisement. (iii) $\lambda^c = \mathsf{T}$ where T is the first transaction of ₿$_{\mathsf{D}}(D, D, \mathsf{T}', o, v, \mathsf{PartG}, 0)$ where $(\mathsf{T}', o) = txout'(y)$. The compiler parameters are obtained as for Item 3. (iv) $txout$ extends $txout'$ mapping x to $(\mathsf{T}, 0)$, $sechash = sechash'$, and $\kappa = \kappa'$.

(10) $\alpha = \mathsf{A} : x, x'$, where: (i) In $\dot{R}^s$ we find $\langle \mathsf{A}, v\rangle_x$ and $\langle \mathsf{A}, v'\rangle_{x'}$.

(11) $\alpha = join(x, y)$, where: (i) In $\dot{R}^s$ the action α spends $\langle \mathsf{A}, v\rangle_x$ and $\langle \mathsf{A}, v'\rangle_{x'}$ to obtain $\langle \mathsf{A}, v + v'\rangle_y$. (ii) $\lambda^c = \mathsf{T}$ is a transaction having as inputs $txout'(x)$ and $txout'(x')$, and having one output of value $v + v'$ redeemable with $\hat{K}_{\mathsf{A}}(r_{\mathsf{A}})$. (iii) $txout$ extends $txout'$ mapping y to $(\mathsf{T}, 0)$, $sechash = sechash'$, and $\kappa = \kappa'$.

(12) $\alpha = \mathsf{A} : x, v, v'$. Similar to Item 10.

(13) $\alpha = divide(x, v, v')$. Similar to Item 11.

(14) $\alpha = \mathsf{A} : x, \mathsf{B}$. Similar to Item 10.

(15) $\alpha = donate(x, \mathsf{B})$. Similar to Item 11.

(16) $\alpha = \mathsf{A} : \vec{y}, j$, where: (i) $\vec{y} = y_1 \cdots y_k$. (ii) In $\dot{R}^s$ we find $\langle \mathsf{B}_i, v_i\rangle_{y_i}$ for $i \in 1..k$, with $\mathsf{B}_j = \mathsf{A}$. (iii) In $\dot{R}^c$ we find $\mathsf{B} \to * : \mathsf{T}$ for some B, T, where T has as its inputs $txout'(y_i)$ for $i \in 1..k$, and possibly others not in ran $txout'$. (iv) $\lambda^c = \mathsf{B} \to * : m$ from some B, m where m is a signature of T with $\hat{K}_{\mathsf{A}}(r_{\mathsf{A}})$, corresponding to the j-th input. (v) λ^c is the first broadcast of m in $\dot{R}^c$ after the first broadcast of T. (vi) λ^c does not correspond to any of the other cases, i.e. there is no other symbolic action α for which $\dot{R}^s\alpha$ would be coherent with $\dot{R}^c\lambda^c$. (vii) $txout = txout'$, $sechash = sechash'$, and $\kappa = \kappa'$.

(17) $\alpha = destroy(\vec{x})$, where: (i) $\vec{x} = x_1 \cdots x_k$. (ii) In $\dot{R}^s$, α consumes $\langle \mathsf{A}_i, v_i\rangle_{x_i}$ to obtain 0. (iii) $\lambda^c = \mathsf{T}$ from some T having as inputs $txout'(x_1), \ldots, txout'(x_k)$, and possibly others not in ran $txout'$. (iv) λ^c does not correspond to any of the other cases, i.e. there is no other symbolic action α for which $\dot{R}^s\alpha$ would be coherent with $\dot{R}^c\lambda^c$. (v) $txout = txout'$, $sechash = sechash'$, and $\kappa = \kappa'$.

(18) $\alpha = \delta = \lambda^c$, and $txout = txout'$, $sechash = sechash'$, and $\kappa = \kappa'$.

Inductive case 2: the predicate $coher(R^s, R^c\lambda^c, r, txout, sechash, \kappa)$ holds if $coher(R^s, R^c, r, txout, sechash, \kappa)$, and one of the following cases applies:

(1) $\lambda^c = \mathsf{T}$ where no input of T belongs to ran $txout$.

(2) $\lambda^c = \mathsf{A} \to \mathsf{O} : m$ or $\lambda^c = \mathsf{O} \to \mathsf{A} : m$, for some A, m.

(3) $\lambda^c = \mathsf{A} \to * : m$, where λ^c does not correspond to any symbolic move, according to the first inductive case.

We write $R^s \sim_r R^c$ iff $coher(R^s, R^c, r, txout, sechash, \kappa)$ for some $txout$, $sechash$, and κ. ⋄

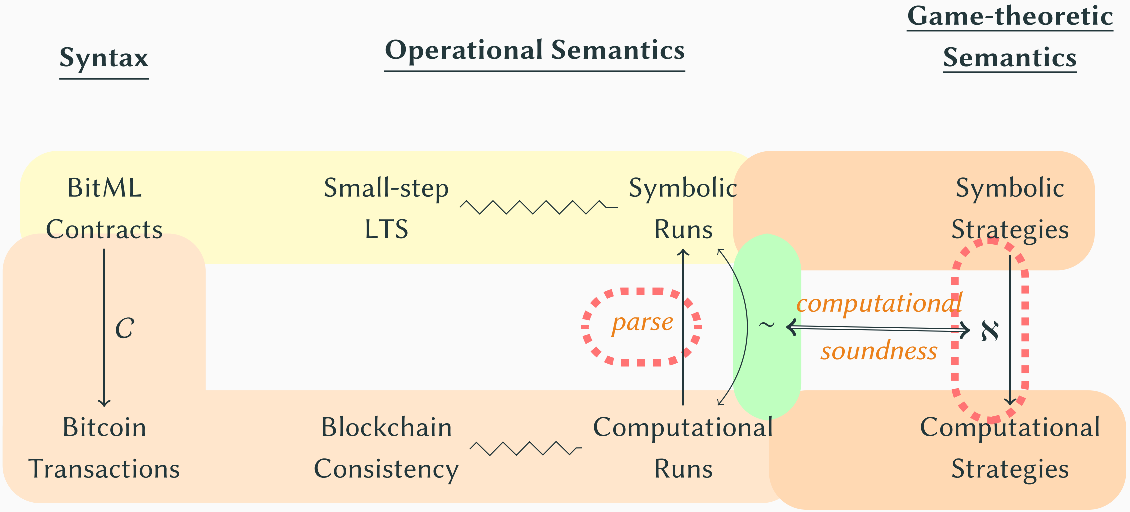
The following lemma is the active contracts analogous of Lemma 1. Both results are proved by induction on the definition of coherence.

Lemma 6. *Let $coher(R^s, R^c, r, txout, sechash, \kappa)$. For each active contract $\langle C, v\rangle_x$ occurring in Γ_{R^s}, there exists a corresponding unspent transaction output (T, o) in $\mathbf{B}_{R^c}$ with value v. Further, T is generated by the invoking the compiler as ₿$_{\mathsf{C}}(C, D_p, \mathsf{T}', o', v, I, \mathcal{P}, t)$ for some values of $D_p, \mathsf{T}', o', I, \mathcal{P}, t$, or as ₿$_{\mathsf{par}}(\vec{C}, D_p, \mathsf{T}', o', \vec{v}, \mathcal{P}, t)$ for some*

Technical challenges for coherence

- Proofs that construct mappings required meta-properties on lists
- Tracing a contract's lifetimer required temporal hyper-properties
- Scaling up → hit Agda's type-checking performance limits

BitML



Syntax
Operational Semantics
Game-theoretic Semantics
BitML Contracts
Small-step LTS
Symbolic Runs
Symbolic Strategies
C
parse
~
computational soundness
ℵ
Bitcoin Transactions
Blockchain Consistency
Computational Runs
Computational Strategies

BitML paper: computational soundness

Using such maps, we can detect when a transaction T in $R^c_\star$ has some input with a symbolic counterpart (i.e. in ran *txout*). When that happens, we can map T into its corresponding action in $R^s_\star$. According to Definitions 16 and 17, T has to be preceded by the broadcasts of its witnesses w, which, in turn, must be preceded by the broadcast of T. This allows to parse the signatures in w, so to generate authorizations in $R^s_\star$.

A few cases must be handled with more care. For instance, Adv could broadcast a signature *before* the signed transaction. When this happens, we simply ignore the message; duplicate signatures are ignored as well. Further, a computational contract advertisement could involve only dishonest participants: we ignore that as well. Adv can consume her own deposits (among those tracked by *txout*), to create an arbitrary transaction without a symbolic counterpart. In this case, in the semantics we use the [Dep-Destroy] move in $R^s_\star$, preceded by its authorizations, making those deposits disappear from the symbolic world. When the hash of a secret is committed, it can not be parsed precisely as a [C-AuthCommit] move, since the latter involves the length of the secrets, which can not be inferred from the hash. However, this is not needed, since the *stripped* run $R^s_\star$ does not involve such lengths.

Definition 22 (From symbolic to computational strategies). Let Σ^s_{A} be a symbolic strategy, with $\mathsf{A} \in \mathsf{Hon}$. We define $\aleph(\Sigma^s_{\mathsf{A}}) = \Sigma^c_{\mathsf{A}}$ below. Given the parameters $R^c_\star$, r_{A} of Σ^c_{A}, we:

(1) parse the (stripped) run $R^c_\star$, so to obtain a corresponding symbolic (stripped) run $R^s_\star$, as sketched above;
(2) halve the random sequence r_{A} as $(\pi_1(r_{\mathsf{A}}), \pi_2(r_{\mathsf{A}}))$;
(3) evaluate $\Lambda^s = \Sigma^s_{\mathsf{A}}(R^s_\star, \pi_1(r_{\mathsf{A}}))$;
(4) convert the symbolic actions Λ^s into computational actions Λ^c, and define $\Sigma^c_{\mathsf{A}}(R^c_\star, r_{\mathsf{A}}) = \Lambda^c$. When Λ^s contains $\mathsf{A} : \{G\}C, \Delta$ or $\mathsf{A} : \{G\}C, x$, their conversion follows the stipulation protocol (Definition 21), using $\pi_2(r_{\mathsf{A}})$. There, at item 5, we choose the length of each secret by adding η to the corresponding value N in Δ. ⋄

A.8 Supplementary material for Section 9

Proof of Theorem 2 Assume that R^c satisfies the hypotheses, but has no corresponding R^s which is coherent (to R^c) and conforming (to the symbolic strategies). Consider the longest prefix $\dot{R}^c$ of R^c having a corresponding $\dot{R}^s$ which is coherent (to $\dot{R}^c$) and conforming (to the computational strategies). We have that $R^c = \dot{R}^c \lambda^c \cdots$. We now show that either $\dot{R}^c \lambda^c$ has a corresponding symbolic run $\dot{R}^s \ddot{R}^s$ which is coherent and conforming to the symbolic strategies (contradicting the maximality of $\dot{R}^c$), or the adversary succeeded in a signature forgery, or in a preimage attack (which can happen only with negligible probability). Note that, by obtain coherence, $\ddot{R}^s$ must be either empty, or contain a single symbolic action.

We proceed by cases on λ^c:

(1) $\lambda^c = \mathsf{B} \to * : m$. Then, coherence must hold for some $\ddot{R}^s$. Indeed, the definition of coherence maps m to an authorization (if it is the first broadcast of a signature), a revealed secret (if it is the first broadcast of a preimage), or in *all* other cases it simply ignores m. So, we can choose $\ddot{R}^s$ as the corresponding move, or to be empty, and obtain coherence

last case ($\ddot{R}^s$ empty) the run $\dot{R}^s \ddot{R}^s = \dot{R}^s$ is trivially conforming. For the authorization or reveal cases, we note that if the computational Adv was able to generate m, it is either forged (with negligible probability), or it originated from some honest A. Since $\Sigma^c_{\mathsf{A}} = \aleph(\Sigma^s_{\mathsf{A}})$, it follows that, at some time in the past, Σ^s_{A} enabled the authorization or reveal.
By persistency, it is also enabled at the end of $\dot{R}^s$, hence Σ^s_{Adv} can choose such action, and achieve conformance.

(2) If $\lambda^c = \mathsf{T}$, we consider the following subcases according to the inputs of T:
   (a) If no input of T belongs to ran *txout*, then coherence and conformance are achieved taking $\ddot{R}^s$ to be empty.
   (b) Otherwise, if at least one of the inputs of T belongs to ran *txout*, then we look in $\dot{R}^s$ for all the deposits and active contracts corresponding to such inputs. By definition of computational strategy, we must find in $\dot{R}^c$ a (first) broadcast $\mathsf{B} \to * : \mathsf{T}$ followed by a (first) broadcast $\mathsf{B} \to * : m$ for all witnesses m of T. By the coherence of $\dot{R}^c$, in $\dot{R}^s$ the messages $\mathsf{B} \to * : m$ correspond to suitable authorization/reveal moves for each of the (counterparts of the) inputs of T. We consider the following subcases:
      (i) If all the inputs are deposits, then we let $\ddot{R}^s$ perform the symbolic move corresponding to T (e.g., init or join). Note that if T can not be represented symbolically, we can choose $\ddot{R}^s$ to perform a destroy. Such moves are feasible symbolically since we already have their authorizations. Such $\ddot{R}^s$ leads to a coherent run, which is also conforming, since even if no honest strategy wants to perform the move, Adv can perform it on its own, having all the authorizations.
      (ii) Otherwise, some input T′ of T must correspond to an active contract. This must be originated from an advertisement, which has to involve at least one honest participant A, by definition of the symbolic semantics (rule [C-Advertise]). However, by construction, our compiler makes T′ require the signatures of all the participants, hence including A. Since such signature must occur as a witness in T, the adversary Adv must have forged it (with negligible probability), or must have obtained it from A. In the latter case, $\dot{R}^s$ contains an authorization for the symbolic move corresponding to λ^c. By choosing $\ddot{R}^s$ accordingly, we obtain a coherent and conforming run.

(3) Finally, if $\lambda^c = \delta$, we simply choose $\ddot{R}^s$ to perform δ. Coherence trivially holds. For conformance, we note that by definition of computational strategy, all the honest participants must output a Λ^c which either contains some $\delta' \geq \delta$, or is empty. By definition of $\aleph$, this must also be the case in Λ^s, resulting in conformance. □

BitML: Factoring out the non-game-theoretic part of the final theorem

Given a computational run R^c and symbolic strategies σ^s:

$$\frac{R^c \quad \text{conforms to } \Sigma^c \quad \text{where } \Sigma^c := \aleph(\Sigma^s)}{\exists R^s.\ R^s \sim R^c \quad R^s \text{ conforms to } \Sigma^s}$$

BACKUP PLAN

Formalize only the first half of *computational soundness* (i.e. parsing).

επιστήμη