Secure Concurrent Constraint Programming

H. Lopez, J. Perez, C. Rueda,
*Universidad Javeriana*

C. Palamidessi
*INRIA and LIX Ecole Polytechnique*

F. Valencia
*CNRS and LIX Ecole Polytechnique.*

**ICLP**, 2006
Motivation

The study of security uses tools and techniques from constraint and logic programming.

- **CP and LP Programming Languages.**
  - Millen’s Interrogator Prolog constraint solver [Mil95].

- **Logic Programming.**
  - Compton et al [CD99] used LP for protocols.
  - Blanchet uses Horn-Clauses for analyzing protocols.

- **Constraint Programming:** Soft Constraints by Bistarelli et al [BB01].
Motivations

Want to explore *Concurrent Constraint Programming* (CCP), a well-established concurrency framework, for reasoning about Security Protocols.

Reasons:

- *Striking resemblance* with modern calculi for security.
- Matured Model with several *reasoning techniques* (specially reachability analysis, traces) and implementations
  - Final-store denotational semantics,
  - Complete proof systems for verification $P \models F$
  - Declarative methods: Processes-as-FOL formulae.
  - Decidability Results for Reachability
  - Programming Languages.
In this talk we report on our observations and preliminary work on CCP as a framework for security protocols.

1. Security Protocols


3. CCP

4. Similarities between CCP and Modern Process Calculi for Security

5. CCP Extension to model Security Protocols

6. Conclusion
Background: Security protocols

- Message exchange via untrusted network
- Achieve authentication, secrecy, anonymity, etc.
- Needham-Schröder protocol

\[ A \rightarrow B : \{m, A\}_{Pub(B)} \]
\[ B \rightarrow A : \{m, n\}_{Pub(A)} \]
\[ A \rightarrow B : \{n\}_{Pub(B)} \]

Two roles: initiator (A), responder (B)

Concurrent sessions among distributed agents

Uses cryptography to achieve security properties

- Secrecy property:
  \( n \) never appears on a run of the protocols.
The Dolev-Yao model Active intruder:

- Eavesdrop
- Disassemble and compose
- Encrypt and decrypt with available keys

BUT

- random numbers unguessable
- cryptography unbreakable

Properties can be difficult to prove
Undecidability of secrecy (even for class of simple prot.)
Security protocol verification

Do it formally:

• 1) protocol formalisation,
• 2) property formalisation,
• 3) the proof (automatic or by hand)

An important approach within Concurrency Theory:

• Process Calculi (enriched with crypto primitives): Treat (concurrent) processes much like the lambda calculi treat computable functions.
Example: SPL

Syntax:

Name expr. \( v \) ::= \( n, A, \cdots \mid x, X, \cdots \)

Key expr. \( k \) ::= \( \text{Pub}(v) \mid \text{Priv}(v) \mid \text{Key}(\overline{v}) \)

Messages \( M \) ::= \( v \mid k \mid M, M' \mid \{M\}_k \mid \psi \)

Processes \( p \) ::= \( \text{out} \ new \overline{x} \ M.p \mid \text{in} \ pat \overline{x} \overline{\psi} \ M.p \mid ||_{i \in I} p_i \)

Convention: \( \text{nil} \equiv ||_{i \in \emptyset} p_i \)
SPL Modelling

Alice and Bob and Dole-Yao Spy:

\[
\begin{align*}
\text{Init}(A, B) & \equiv \text{out new } x \{x, A\}_{\text{Pub}(B)} \cdot \\
& \quad \text{in } \{x, y, B\}_{\text{Pub}(A)} \cdot \\
& \quad \text{out } \{y\}_{\text{Pub}(B)} \cdot \\
& \quad \text{nil}
\end{align*}
\]

\[
\begin{align*}
\text{Resp}(B) & \equiv \text{in } \{x, Z\}_{\text{Pub}(B)} \cdot \\
& \quad \text{out new } y \{x, y, B\}_{\text{Pub}(Z)} \cdot \\
& \quad \text{in } \{y\}_{\text{Pub}(B)} \cdot \\
& \quad \text{nil}
\end{align*}
\]

\[
\begin{align*}
\text{Spy}_1 & \equiv \text{in } \psi_1 . \text{in } \psi_2 . \text{out } \psi_1 . \psi_2 . \text{nil} & \text{(composing)} \\
\text{Spy}_2 & \equiv \text{in } \psi_1 , \psi_2 . \text{out } \psi_1 . \text{out } \psi_2 . \text{nil} & \text{(decomposing)} \\
\text{Spy}_3 & \equiv \text{in } x . \text{in } \psi . \text{out } \{\psi\}_{\text{Pub}(x)} \cdot \text{nil} & \text{(encrypting)} \\
\text{Spy}_4 & \equiv \text{in } \text{Priv}(x) . \text{in } \{\psi\}_{\text{Pub}(x)} \cdot \text{out } \psi \cdot \text{nil} & \text{(decrypting)} \\
\text{Spy} & \equiv \|_{i \in \{1, \ldots, 4\}} \text{Spy}_i
\end{align*}
\]

The System Modelling All Attacks:

\[
\begin{align*}
\text{P}_{\text{init}} & \equiv \|_{A, B} ! \text{Init}(A, B) \\
\text{P}_{\text{resp}} & \equiv \|_{A} ! \text{Resp}(A) \\
\text{P}_{\text{spy}} & \equiv ! \text{Spy} \\
\text{NSL} & \equiv \|_{i \in \{\text{resp, init, spy}\}} \text{P}_i
\end{align*}
\]
SPL Monotonic Semantics

Semantics: < Process, Names, Store >

(output) Provided the names \( \vec{n} \) are all distinct and not in \( s \),

\[
\langle \text{out new } \vec{x} M . p, s, t \rangle \xrightarrow{\text{out new } \vec{n} M[\vec{n}/\vec{x}]} \langle p[\vec{n}/\vec{x}], s \cup \{ \vec{n} \}, t \cup \{ M[\vec{n}/\vec{x}] \} \rangle
\]

(input) Provided \( M[\vec{n}/\vec{x}, \vec{N}/\vec{\psi}] \in t \),

\[
\langle \text{in } \text{pat } \vec{x} \vec{\psi} M . p, s, t \rangle \xrightarrow{\text{in } \text{pat } \vec{x} \vec{\psi} M[\vec{n}/\vec{x},\vec{N}/\vec{\psi}]} \langle p[\vec{n}/\vec{x}, \vec{N}/\vec{\psi}], s, t \rangle
\]

(par)

\[
\langle p_j, s, t \rangle \xrightarrow{\alpha} \langle p'_j, s', t' \rangle \quad \langle \text{par } p_i \rangle \xrightarrow{j: \alpha} \langle \text{par } p_i[p'_j/j], s', t' \rangle
\]  

Remember: **Monotonic Store Assumption** and **Name Generation**
Example: Boreale and Buscemi’s

Syntax:

\[ M, N ::= x \mid a \mid \langle M, N \rangle \mid \{M\}_k \]

\[ P, Q ::= 0 \mid a(x).P \mid \overline{a}(M).P \mid [M = N]P \mid \text{case } M \text{ of } \{y\}_k \text{ in } P \mid \text{pair } M \text{ of } \langle x, y \rangle \text{ in } P \mid (\nu b)P \mid P \mid Q \]

Message Inference System:

\[
\begin{array}{c}
M \in S \\
\hline
S \vdash M
\end{array}
\quad
\begin{array}{c}
S \vdash M \\
S \vdash k
\end{array}
\quad
\begin{array}{c}
S \vdash \{M\}_k \\
S \vdash k
\end{array}
\quad
\begin{array}{c}
S \vdash M \\
S \vdash \{M\}_k \\
S \vdash k
\end{array}
\]

\[ S \vdash M \]
Boreale and Buscemi’s Semantics

**Semantics** (Configurations Store ▷ Process).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(INP)</td>
<td>$s \triangleright a(x).P \quad \rightarrow \quad s \cdot a(M) \triangleright P^{[M/x]}$ where $s \vdash M$</td>
</tr>
<tr>
<td>(OUT)</td>
<td>$s \triangleright a(M).P \quad \rightarrow \quad s \cdot a(M) \triangleright P$</td>
</tr>
<tr>
<td>(CASE)</td>
<td>$s \triangleright \text{case } {M}<em>{k} \text{ of } {y}</em>{k} \text{ in } P \quad \rightarrow \quad s \triangleright P^{[M/y]}$</td>
</tr>
<tr>
<td>(SPLIT)</td>
<td>$s \triangleright \langle M, N \rangle \text{ of } \langle x, y \rangle \text{ in } P \quad \rightarrow \quad s \triangleright P^{[M/x, N/y]}$</td>
</tr>
<tr>
<td>(MATCH)</td>
<td>$s \triangleright [M = M]P \quad \rightarrow \quad s \triangleright P$</td>
</tr>
<tr>
<td>(RES)</td>
<td>$s \triangleright (\nu a)P \quad \rightarrow \quad s[a'/a] \triangleright P^{[a'/a]}$ where $a'$ is fresh for $s, P$</td>
</tr>
<tr>
<td>(PAR)</td>
<td>$s \triangleright P \quad \rightarrow \quad s' \triangleright P'$</td>
</tr>
</tbody>
</table>

$s \triangleright P \mid Q \quad \rightarrow \quad s' \triangleright P' \mid Q$

**Remember:** Monotonic Store, Parametric in Inference System, and Name Generation.
CCP Scenario & Concurrency

- Concurrent Executions of Agents.
- Synchronization: Via Blocking-Ask.
CCP Syntax

Let $x, y, \ldots$, be variables and $\gamma, \sigma, \ldots$ be constraints (i.e., FOL formulae)

Agents $A, B, \ldots := \text{true} \mid \text{tell}(\sigma) \mid \text{ask}(\sigma).A \mid (\nu x).A \mid A \parallel B \mid p(X)$

Definitions $D := \epsilon \mid p(X) :: A \mid D.D$

Programs $P, Q := D.A$
Operational Semantics

\[ A^{(s,s')} \rightarrow B \] means: \( A \) in store \( s \) reduces to \( B \) in \( s' \).

**TELL:**
\[ \text{tell}(\sigma)^{(s,s\wedge\sigma)} \rightarrow \text{true} \]

**ASK:**
\[ \text{ask}(\sigma).A^{(s,s)} \rightarrow A \text{, if } s \vdash \sigma \]

**PAR:**
\[ \frac{A^{(s,s')} \rightarrow A'}{A \parallel B^{(s,s')} \rightarrow A' \parallel B} \]

**HIDE:**
\[ \frac{A^{(s,s')} \rightarrow A'}{(\text{local}X)A^{(s,s')} \rightarrow A'} \text{ if } X \text{ is fresh} \]
A program example

\[
D = \quad \max(X, Y, Z) \implies (\text{ask}(X \geq Y).\text{tell}(Z = X)) \nonumber \\
\quad \quad \| (\text{ask}(Y > X).\text{tell}(Z = Y)) \nonumber 
\]

\[
A_1 = \quad \text{tell}(X > 20) \| \text{tell}(Y > 10) \| \max(X, Y, Z) 
\]

\[
A_2 = \quad \text{tell}(X > 20) \| \text{tell}(Y < 10) \| \max(X, Y, Z) 
\]

- In \( A_1 \) (initiated in the empty store) both ask agents are stuck.

- In \( A_2 \) (initiated in any store) \( \text{tell}(Z = X) \) is output.
Constraint Systems

CCP is parametric in a constraint system

**Definition.** A constraint system consists of a signature \( \Sigma \) and first-order theory \( \Delta \) over \( \Sigma \).

- **Constraints**: formulae over \( \Sigma \).
- **Relation** \( \vdash \): decidable entailment relation between constraints.

**Example:** Boreale-Buscemi’s constraint system: \( \Sigma \) specifies the crypto primitives and \( \vdash \) is given by

\[
\begin{align*}
M \in S & \quad S \vdash M & \quad S \vdash k \\
\overline{S \vdash M} & \quad \overline{S \vdash \{M\}_k} \\
S \vdash \{M\}_k & \quad S \vdash k & \quad S \vdash M
\end{align*}
\]
Observations

**Observation 1:** Boreale’s Inference System \( \vdash \) gives rise to a particular constraint system.

**Observation 2:** CCP has a Monotonic Store (like SPL, Boreale’s calculus, and others)
Some Reasoning Techniques

Closure-Operator and Logic Interpretations

Denotation \([ \cdot ]\)
\[
\begin{align*}
[\text{tell}(\sigma)] &= \{ \gamma \mid \gamma \vdash \sigma \} \\
[\text{ask}(\sigma).A] &= \{ \gamma \mid \gamma \vdash \sigma \Rightarrow \gamma \in [A] \} \\
[\nu x A] &= \{ \gamma \mid \exists \sigma \in [A] : \exists x \sigma \Leftrightarrow \exists x \gamma \} \\
[A \parallel B] &= [A] \cap [B]
\end{align*}
\]

Formula \(F(\cdot)\)
\[
\begin{align*}
\sigma \\
\sigma \Rightarrow F(A) \\
\exists x F(A) \\
F(A) \land F(B)
\end{align*}
\]

Remark 1. (Behavioral Characterization)
\([A] = [B]\) iff not context can operationally distinguish \(A\) from \(B\).

Remark 2. (Reachability Characterization)
\(A\) output \(\sigma\) iff and only if \(F(A) \Rightarrow \sigma\)

Notice the correspondence between \((\nu x)A\) and \(\exists x F(A)\).

Other tools: Proof systems for Model Checking, Petri Net Semantics, True Concurrent Semantics.
Modelling Security Protocols with CCP

A CCP Strategy for reasoning about protocols:

- Use CCP with Boreales’s constraint system.
- Use Tells to output messages on the network and Asks to input them.
- Then use reasoning techniques of CCP.

Almost done. Do we have name (or nonce) generation and name passing as most of the calculi for security?

- Can the CCP “(νx)”, i.e., existential quantification “(∃x)”, simulate the Pi-calculus (νx) ?

Not quite. But we will see to it.
Modelling the Pi ($\nu x$) in CCP

- Assume a constraint system with only predicates of the form $out(.)$. The process

$$tell(out(42)) \parallel ask(out(y)).A.$$ 

does not execute $A$.

- Let us then extend the Ask to Persistent Universal Ask:

$$B = \forall \vec{y} \ ask(\sigma[\vec{y}]).A$$

Operationally:

$$\begin{align*}
B & \xrightarrow{(s,s)} A[\vec{x}/\vec{y}] \parallel B, \text{ if } s \vdash \sigma[\vec{x}/\vec{y}] .
\end{align*}$$

- The logic counter-part is $\forall \vec{y} \ \sigma[\vec{y}] \Rightarrow F(A)$.

- Now we have name passing. E.g., the process

$$tell(out(42)) \parallel \forall y \ ask(out(y)).A.$$ 

Modelling the Pi ($\nu x$) in CCP

Do we have name generation? Can we now capture the pi-calculus $\nu$ with $\exists$?

Consider the interpretation of a replicated version of the $\pi$ calculus in the logical counter-part of our extended CCP:

\[
\begin{align*}
[0] &= \text{true} \\
[!x(z)] &= \text{out}(x, z) \\
![x(y).P] &= \forall y(\text{out}(x, y) \Rightarrow [P]) \\
[P \mid Q] &= [P] \land [Q] \\
[(\nu x)P] &= \exists x[P]
\end{align*}
\]

**Remark 1**: $\pi$ reduction as logic entailment.

\[\text{if } P \rightarrow Q \text{ then } [P] \models [Q]\]

**Remark 2**: FOL Characterization of $\pi$ Barb Observability.

\[P \Downarrow x \text{ if and only if } [P] \models \exists z \text{ out}(x, z)\]
Modelling the Pi ($\nu x$) in CCP

• Hence if we are interested in certain kind of reachability analysis $\exists$ can capture the $\nu$ operator of the $\pi$-calculus.

• Application: Identification of meaningful $\pi$-classes for which barbed reachability is decidable by using Classic results from FOL. E.g.,

1. $\{ R \mid R \Rightarrow \forall x \exists y F \}$ (Bernays-Schönfinkel’s class).
2. $\{ R \mid R \Rightarrow \forall x \exists u w \forall y F \}$ (Gödel’s class).
3. $\{ R \mid R \equiv R’ \text{ for some } R’ \text{ s.t. } |fn(R’) \cup bn(R’)| \leq 2 \}$ (Two-Variables Class).

Example: Berney-Schönfinkel’s process $!x(y). (\nu z)!\bar{y}z$ is a provider of fresh names. The Goedel’s process $(\nu z)(!\bar{x}z \mid !z(u).\bar{u})$ waits on its secret channel.
Modelling the Pi \((\nu x)\) in CCP

**Example:** Take \(P = (\nu z)(!\bar{x}z \mid !z(u).\bar{u})\) and \(Q = x(y).\bar{y}t\) and \(R = Q \parallel P\). We have \(R \Downarrow t\).

From the translation we get:

\[
[R] = \forall y (\text{out}(x, y) \Rightarrow \text{out}(y, t)) \\
\quad \land \exists z (\text{out}(x, z) \land \forall u \text{out}(z, u) \Rightarrow \text{out}(u)).
\]

which is logically equivalent to

\[
F = \exists z (\forall y (\text{out}(x, y) \Rightarrow \text{out}(y, t)) \\
\quad \land \text{out}(x, z) \land \forall u \text{out}(z, u) \Rightarrow \text{out}(u)).
\]

Now, since \(\text{out}(z, t)\) is a logical consequence of \(\forall y (\text{out}(x, y) \Rightarrow \text{out}(y, t)) \land \text{out}(x, z)\), we have \(F \models \exists z \text{out}(z, t) \land \forall u \text{out}(z, u) \Rightarrow \text{out}(u)\) from which we obtain \(F \models \text{out}(t)\).
Modelling the Pi ($\nu x$) in CCP

But how robust is the interpretation of $\nu$ as $\exists$?

The correspondence holds in the absence of mismatch. E.g., it would not hold for mismatch $[x \neq y].P$ with the natural interpretation $x \neq y \Rightarrow \llbracket P \rrbracket$.

Let $R = (\nu x)(\nu y)[x \neq y].!\bar{t}$.

- We have $R \Downarrow !\bar{t}$
- But $\llbracket R \rrbracket = \exists x \exists y (x \neq y \Rightarrow \text{out}(t)) \not\vdash \text{out}(t)$.

So if want a more faithfull correp ondance we may have to consider operators to reason about freshness, uniqueness, etc. E.g., Miller’s $\nabla$ operator.
Conclusions

We believe that CCP + Universal Persistent Asks may provide a framework for reasoning about security protocols:

- *Constraint System* specifies crypt primitives and spy knowledge.

- *Monotonic store* models the power of the spy (see and remember everything)

- *Existential and Universal* model nonce generation and name passing.

- The non-linear (i.e., persistent) nature of the framework makes it suitable for protocols analysis with unbounded runs.
References

