Spatial and Epistemic Modalities in Constraint-Based Process Calculi

S. Knight (INRIA) C. Palamidessi (INRIA)
P. Panangaden (McGill Univ.) F. Valencia (CNRS.)

HIGHLIGHTS 2013

Distributed Systems with Spatial Hierarchies

- Our goal: a process calculus that can express information in multi-agent distributed systems.
- Distributed Systems (DS) have changed substantially due to social networks and cloud computing.
- Agents post and share partial information and programs in a cloud with spatial hierarchies.
- Example: Posting information in friend circles, sharing albums and apps on Facebook.

Distributed Systems with Spatial Hierarchies

- Our goal: a process calculus that can express information in multi-agent distributed systems.
- Distributed Systems (DS) have changed substantially due to social networks and cloud computing.
- Agents post and share partial information and programs in a cloud with spatial hierarchies.
- Example: Posting information in friend circles, sharing albums and apps on Facebook.

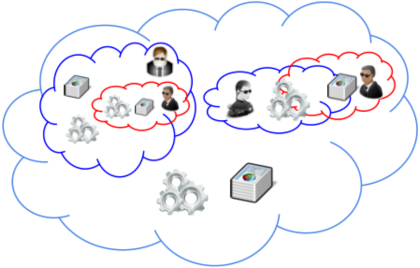
Distributed Systems with Spatial Hierarchies

- Our goal: a process calculus that can express information in multi-agent distributed systems.
- Distributed Systems (DS) have changed substantially due to social networks and cloud computing.
- Agents post and share partial information and programs in a cloud with spatial hierarchies.
- Example: Posting information in friend circles, sharing albums and apps on Facebook.

Distributed Systems with Spatial Hierarchies

- Our goal: a process calculus that can express information in multi-agent distributed systems.
- Distributed Systems (DS) have changed substantially due to social networks and cloud computing.
- Agents post and share partial information and programs in a cloud with spatial hierarchies.
- Example: Posting information in friend circles, sharing albums and apps on Facebook.

Distributed Systems with Spatial Hierarchies



Typically, within their spaces in the cloud, agents (users) may:

- run applications,
- post and ask local information (possibly inconsistent).
- announce and ask facts (global information, knowledge).

So we are interested in distributed systems exhibiting: nesting of spaces, local failure, locality and globality.

Our Aim

A model with the emphasis on representing and managing access to information in distributed systems.

- Posting and querying information and knowledge within the spatial hierarchies.
- Running processes within the spatial hierarchies.

We propose a framework:

- ***Process Calculi with Epistemic and Spatial Modalities.***

Approach: Spatial/Epistemic CCP.

We build upon the process calculus CCP, because it is closely linked to logic, and it deals with partial information in a constraint system.

We generalize the theory of CCP with:

- A general domain-theoretical notion of spatial and epistemic constraint systems.
- A distributed hierarchical store.
- A spatial construction specifying processes computing with information within a space.
- An epistemic construction specifying processes computing with knowledge within a space.

Approach: Spatial/Epistemic CCP.

We build upon the process calculus CCP, because it is closely linked to logic, and it deals with partial information in a constraint system.

We generalize the theory of CCP with:

- A general domain-theoretical notion of spatial and epistemic constraint systems.
- A distributed hierarchical store.
- A spatial construction specifying processes computing with information within a space.
- An epistemic construction specifying processes computing with knowledge within a space.

Constraint Systems

A constraint system is a lattice of partial information with a notion of entailment ($\sqsubseteq$) and an operation for joining pieces of information ($c \sqcup d$).

Our constraint systems are called **Spatial Constraint Systems.** Agents may have their own local spaces. In our approach *each agent i has a space function $\mathfrak{s}_i : Con \rightarrow Con$* in the cs.

- Locality: $\mathfrak{s}_i(c)$ means that c holds in the space attributed to agent i.
- Nesting: $\mathfrak{s}_i(\mathfrak{s}_j(c))$ means that c holds within a space that i attributes to j. Nesting can be of any depth.

Constraint Systems

A constraint system is a lattice of partial information with a notion of entailment ($\sqsubseteq$) and an operation for joining pieces of information ($c \sqcup d$).

Our constraint systems are called **Spatial Constraint Systems.** Agents may have their own local spaces. In our approach *each agent i has a space function $\mathfrak{s}_i : Con \to Con$* in the cs.

- Locality: $\mathfrak{s}_i(c)$ means that c holds in the space attributed to agent i.
- Nesting: $\mathfrak{s}_i(\mathfrak{s}_j(c))$ means that c holds within a space that i attributes to j. Nesting can be of any depth.

Constraint Systems

A constraint system is a lattice of partial information with a notion of entailment ($\sqsubseteq$) and an operation for joining pieces of information ($c \sqcup d$).

Our constraint systems are called **Spatial Constraint Systems.** Agents may have their own local spaces. In our approach *each agent i has a space function $\mathfrak{s}_i : Con \to Con$* in the cs.

- Locality: $\mathfrak{s}_i(c)$ means that c holds in the space attributed to agent i.
- Nesting: $\mathfrak{s}_i(\mathfrak{s}_j(c))$ means that c holds within a space that i attributes to j. Nesting can be of any depth.

Constraint Systems

A constraint system is a lattice of partial information with a notion of entailment ($\sqsubseteq$) and an operation for joining pieces of information ($c \sqcup d$).

Our constraint systems are called **Spatial Constraint Systems.** Agents may have their own local spaces. In our approach *each agent i has a space function $\mathfrak{s}_i : Con \to Con$* in the cs.

- Locality: $\mathfrak{s}_i(c)$ means that c holds in the space attributed to agent i.
- Nesting: $\mathfrak{s}_i(\mathfrak{s}_j(c))$ means that c holds within a space that i attributes to j. Nesting can be of any depth.

Spatial Constraint Systems

Our requirements:

- Truth: We require $\mathfrak{s}_i(true) = true$ meaning that having no information in a local store amounts to nothing.
- Distribution: We require $\mathfrak{s}_i(c) \sqcup \mathfrak{s}_i(d) = \mathfrak{s}_i(c \sqcup d)$ for joining/distributing local info.

Corollary

If $c \sqsubseteq d$ then $\mathfrak{s}_i(c) \sqsubseteq \mathfrak{s}_i(d)$. I.e., the local spaces are closed under entailment $\sqsubseteq$.

Spatial Constraint Systems

Our requirements:

- Truth: We require $\mathfrak{s}_i(true) = true$ meaning that having no information in a local store amounts to nothing.
- Distribution: We require $\mathfrak{s}_i(c) \sqcup \mathfrak{s}_i(d) = \mathfrak{s}_i(c \sqcup d)$ for joining/distributing local info.

Corollary

If $c \sqsubseteq d$ then $\mathfrak{s}_i(c) \sqsubseteq \mathfrak{s}_i(d)$. I.e., the local spaces are closed under entailment $\sqsubseteq$.

Spatial Constraint Systems

Our requirements:

- Truth: We require $\mathfrak{s}_i(true) = true$ meaning that having no information in a local store amounts to nothing.
- Distribution: We require $\mathfrak{s}_i(c) \sqcup \mathfrak{s}_i(d) = \mathfrak{s}_i(c \sqcup d)$ for joining/distributing local info.

Corollary

If $c \sqsubseteq d$ then $\mathfrak{s}_i(c) \sqsubseteq \mathfrak{s}_i(d)$. I.e., the local spaces are closed under entailment $\sqsubseteq$.

Spatial Constraint Systems

Our requirements:

- Truth: We require $\mathfrak{s}_i(true) = true$ meaning that having no information in a local store amounts to nothing.
- Distribution: We require $\mathfrak{s}_i(c) \sqcup \mathfrak{s}_i(d) = \mathfrak{s}_i(c \sqcup d)$ for joining/distributing local info.

Corollary

If $c \sqsubseteq d$ then $\mathfrak{s}_i(c) \sqsubseteq \mathfrak{s}_i(d)$. I.e., the local spaces are closed under entailment $\sqsubseteq$.

Epistemic Constraint Systems

We now use $\mathfrak{s}_i(c)$ to represent not only a piece of information c that i has but also a fact that he knows.

- $c \sqsubseteq \mathfrak{s}_i(c)$ (i.e. c must be a fact if i knows it)
- $\mathfrak{s}_i(\mathfrak{s}_i(c)) = \mathfrak{s}_i(c)$ (i.e, an agent knows what he knows)
- These requirements mirror the axioms in S4, a variant of epistemic logic.

Fact

Each $\mathfrak{s}_i$ is a Kuratowski closure operator wrt $\sqsubseteq$ (i.e., lub and bottom preserving closure operators).

Epistemic Constraint Systems

We now use $\mathfrak{s}_i(c)$ to represent not only a piece of information c that i has but also a fact that he knows.

- $c \sqsubseteq \mathfrak{s}_i(c)$ (i.e. c must be a fact if i knows it)
- $\mathfrak{s}_i(\mathfrak{s}_i(c)) = \mathfrak{s}_i(c)$ (i.e, an agent knows what he knows)
- These requirements mirror the axioms in S4, a variant of epistemic logic.

Fact

Each $\mathfrak{s}_i$ is a Kuratowski closure operator wrt $\sqsubseteq$ (i.e., lub and bottom preserving closure operators).

Epistemic Constraint Systems

We now use $\mathfrak{s}_i(c)$ to represent not only a piece of information c that i has but also a fact that he knows.

- $c \sqsubseteq \mathfrak{s}_i(c)$ (i.e. c must be a fact if i knows it)
- $\mathfrak{s}_i(\mathfrak{s}_i(c)) = \mathfrak{s}_i(c)$ (i.e, an agent knows what he knows)
- These requirements mirror the axioms in S4, a variant of epistemic logic.

Fact

Each $\mathfrak{s}_i$ is a Kuratowski closure operator wrt $\sqsubseteq$ (i.e., lub and bottom preserving closure operators).

Epistemic Constraint Systems

We now use $\mathfrak{s}_i(c)$ to represent not only a piece of information c that i has but also a fact that he knows.

- $c \sqsubseteq \mathfrak{s}_i(c)$ (i.e. c must be a fact if i knows it)
- $\mathfrak{s}_i(\mathfrak{s}_i(c)) = \mathfrak{s}_i(c)$ (i.e, an agent knows what he knows)
- These requirements mirror the axioms in S4, a variant of epistemic logic.

Fact

Each $\mathfrak{s}_i$ is a Kuratowski closure operator wrt $\sqsubseteq$ (i.e., lub and bottom preserving closure operators).

Epistemic Constraint Systems

We now use $\mathfrak{s}_i(c)$ to represent not only a piece of information c that i has but also a fact that he knows.

- $c \sqsubseteq \mathfrak{s}_i(c)$ (i.e. c must be a fact if i knows it)
- $\mathfrak{s}_i(\mathfrak{s}_i(c)) = \mathfrak{s}_i(c)$ (i.e, an agent knows what he knows)
- These requirements mirror the axioms in S4, a variant of epistemic logic.

Fact

Each $\mathfrak{s}_i$ is a Kuratowski closure operator wrt $\sqsubseteq$ (i.e., lub and bottom preserving closure operators).

Syntax

The syntax is the same for SCCP and ECCP:

$$P, Q, \ldots ::= \mathbf{0} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \rightarrow P \mid P \parallel Q \mid [P]_i \mid X \mid \mu X.P$$

- This is the syntax for traditional CCP with the addition of the $[P]_i$ operator.
- Intuition: $[P]_i$ represents P executing inside agent i's space.

Syntax

The syntax is the same for SCCP and ECCP:

$$P, Q, \ldots ::= \mathbf{0} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \rightarrow P \mid P \parallel Q \mid [P]_i \mid X \mid \mu X.P$$

- This is the syntax for traditional CCP with the addition of the $[P]_i$ operator.
- Intuition: $[P]_i$ represents P executing inside agent i's space.

Syntax

The syntax is the same for SCCP and ECCP:

$$P, Q, \ldots ::= \mathbf{0} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \rightarrow P \mid P \parallel Q \mid [P]_i \mid X \mid \mu X.P$$

- This is the syntax for traditional CCP with the addition of the $[P]_i$ operator.
- Intuition: $[P]_i$ represents P executing inside agent i's space.

Operational Semantics

Spatial Case

The underlying cs must be a spatial cs. Reductions $\langle P, d\rangle \to \langle P', d'\rangle$

$$\mathbf{T}\ \frac{}{\langle \mathbf{tell}(c), d\rangle \longrightarrow \langle \mathbf{0}, d \sqcup c\rangle} \qquad \mathbf{A}\ \frac{c \sqsubseteq d}{\langle \mathbf{ask}\ (c)\ \to\ P, d\rangle \longrightarrow \langle P, d\rangle}$$

$$\mathbf{PL}\ \frac{\langle P, d\rangle \longrightarrow \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \longrightarrow \langle P' \parallel Q, d'\rangle} \qquad \mathbf{R}\ \frac{\langle P[\mu X.P/X], d\rangle \longrightarrow \gamma}{\langle \mu X.P, d\rangle \longrightarrow \gamma}$$

$$\mathbf{S}\ \frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i\, , c\rangle \longrightarrow \langle [P']_i\, , c \sqcup \mathfrak{s}_i(c')\rangle}$$

S Rule and View Operator

- S $$\frac{\langle P, c^i \rangle \longrightarrow \langle P', c' \rangle}{\langle [P]_i, c \rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c') \rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup \{ d \mid \mathfrak{s}_i(d) \sqsubseteq c \}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i \rangle \rightarrow \langle P', c^i \sqcup d \rangle$ then $\langle [P]_i, c \rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d) \rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

S Rule and View Operator

- S $$\frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i, c\rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c')\rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup\{d \mid \mathfrak{s}_i(d) \sqsubseteq c\}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i\rangle \rightarrow \langle P', c^i \sqcup d\rangle$ then $\langle [P]_i, c\rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d)\rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

S Rule and View Operator

- S $$\frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i, c\rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c')\rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup\{d \mid \mathfrak{s}_i(d) \sqsubseteq c\}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i\rangle \rightarrow \langle P', c^i \sqcup d\rangle$ then $\langle [P]_i, c\rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d)\rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

S Rule and View Operator

- S $$\frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i, c\rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c')\rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup\{d \mid \mathfrak{s}_i(d) \sqsubseteq c\}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i\rangle \rightarrow \langle P', c^i \sqcup d\rangle$ then $\langle [P]_i, c\rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d)\rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

S Rule and View Operator

- S $$\frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i, c\rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c')\rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup\{d \mid \mathfrak{s}_i(d) \sqsubseteq c\}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i\rangle \rightarrow \langle P', c^i \sqcup d\rangle$ then $\langle [P]_i, c\rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d)\rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

S Rule and View Operator

- S $$\frac{\langle P, c^i\rangle \longrightarrow \langle P', c'\rangle}{\langle [P]_i, c\rangle \longrightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(c')\rangle}$$
- Definition: Agent i's *view* of c: $c^i = \bigsqcup\{d \mid \mathfrak{s}_i(d) \sqsubseteq c\}$.
- Note: for any constraint c, $c \sqcup \mathfrak{s}_i(c^i) = c$.
- If $\langle P, c^i\rangle \rightarrow \langle P', c^i \sqcup d\rangle$ then $\langle [P]_i, c\rangle \rightarrow \langle [P']_i, c \sqcup \mathfrak{s}_i(d)\rangle$.
- If $\mathfrak{s}_i(c) = \mathfrak{s}_i(d)$ then $(\mathfrak{s}_i(c))^i$ entails both c and d. This is intended: it means that agent i cannot distinguish c from d.
- The operational semantics for ECCP is slightly different. We omit this due to time.

Other Results and Future Work

- We have good reasoning techniques: full abstraction for observational equivalence, barbed equivalence and denotational semantics.
- Future work: develop compelling applications to real-world problems,
- Decidability of the process calculi,
- Model other modal logics, particularly
- Add temporal modalities, to enable fact-changing actions without losing monotonicity.

Other Results and Future Work

- We have good reasoning techniques: full abstraction for observational equivalence, barbed equivalence and denotational semantics.
- Future work: develop compelling applications to real-world problems,
- Decidability of the process calculi,
- Model other modal logics, particularly
- Add temporal modalities, to enable fact-changing actions without losing monotonicity.

Other Results and Future Work

- We have good reasoning techniques: full abstraction for observational equivalence, barbed equivalence and denotational semantics.
- Future work: develop compelling applications to real-world problems,
- Decidability of the process calculi,
- Model other modal logics, particularly
- Add temporal modalities, to enable fact-changing actions without losing monotonicity.

Other Results and Future Work

- We have good reasoning techniques: full abstraction for observational equivalence, barbed equivalence and denotational semantics.
- Future work: develop compelling applications to real-world problems,
- Decidability of the process calculi,
- Model other modal logics, particularly
- Add temporal modalities, to enable fact-changing actions without losing monotonicity.

Other Results and Future Work

- We have good reasoning techniques: full abstraction for observational equivalence, barbed equivalence and denotational semantics.
- Future work: develop compelling applications to real-world problems,
- Decidability of the process calculi,
- Model other modal logics, particularly
- Add temporal modalities, to enable fact-changing actions without losing monotonicity.

Conclusion

We have presented *constraint systems* for agents agents posting and querying information and knowledge in the presence of local, possibly nested, computational spaces.

Thank you.

Conclusion

We have presented *constraint systems* for agents agents posting and querying information and knowledge in the presence of local, possibly nested, computational spaces.

Thank you.