# Computational Fields

## SMuC and CFC, two models compared

Lorenzo Ceragioli

January 15, 2018

# Table of Contents


1. Introduction
2. SMuC
3. CFC
4. Conclusion
5. References

# Introduction

**In physics**: A field is a physical entity that has a value (scalar, vector, tensor ...) for each point in space.

**Examples**

- Temperature field (scalar)
- Gravitational field (vector)

## Field in physics and Computational Fields

A **Computational Field** is a computing system that assign a value to each point in space.

- The space topology is not necessarily regular
- Communication is only local

A **Computational Field** is a computing system that assign a value to each point in space.

- The space topology is not necessarily regular
- Communication is only local

A **Graph based computational field** is a computational field in witch the space topology is defined by a graph structure.

- Values are assigned to nodes
- Edges represent communication capabilities

A **Computational Field** is a computing system that assign a value to each point in space.

- The space topology is not necessarily regular
- Communication is only local

A **Graph based computational field** is a computational field in witch the space topology is defined by a graph structure.

- Values are assigned to nodes
- Edges represent communication capabilities

We want to express the **emergent behavior** of the system

**Soft Mu-Calculus for Computational Fields (SMuC)**

- based on **fixpoints** computation
- the composition of the values propagated by the neighboring nodes is expressed by **mu-calculus-like formulas**
- domains for values of nodes are from **constraint semiring**

# SMuC and CFC

**Soft Mu-Calculus for Computational Fields (SMuC)**

- based on **fixpoints** computation
- the composition of the values propagated by the neighboring nodes is expressed by **mu-calculus-like formulas**
- domains for values of nodes are from **constraint semiring**

**Computational Fields Calculus (CFC)**

- minimal
- it allows the restriction of a field computation to a **sub-region** of the network
- more **simple**

# SMuC and CFC

**Soft Mu-Calculus for Computational Fields (SMuC)**

- based on **fixpoints** computation
- the composition of the values propagated by the neighboring nodes is expressed by **mu-calculus-like formulas**
- domains for values of nodes are from **constraint semiring**

**Computational Fields Calculus (CFC)**

- minimal
- it allows the restriction of a field computation to a **sub-region** of the network
- more **simple**

**Both SMuC and CFC** support the definition of computational field through:

- composition of fields
- propagation of values between neighboring nodes
- the field evolution over time

# SMuC

# Soft Mu-Calculus main ideas

**Fields:** networks with attributes on both nodes and arcs.

# Soft Mu-Calculus main ideas

**Fields:** networks with attributes on both nodes and arcs.

**Execution:** sequential computation of fixpoints.

**Fields:** networks with attributes on both nodes and arcs.

**Execution:** sequential computation of fixpoints.

**Values:** from some *constraint semiring*

# Soft Mu-Calculus main ideas

**Fields:** networks with attributes on both nodes and arcs.

**Execution:** sequential computation of fixpoints.

**Values:** from some *constraint semiring*

- **Robustness against node unavailability**, nodes can proceed with different speed and the result of execution is the same
- **Robustness against node failure**, nodes can fail and return to a precedent consistent state and the result of execution is the same

## Soft Mu-Calculus main ideas

**Fields:** networks with attributes on both nodes and arcs.

**Execution:** sequential computation of fixpoints.

**Values:** from some *constraint semiring*

- **Robustness against node unavailability**, nodes can proceed with different speed and the result of execution is the same
- **Robustness against node failure**, nodes can fail and return to a precedent consistent state and the result of execution is the same

**Distributed implementation** of the calculus is presented

# Constraint semiring

**Field domain** : $\langle A, \sqsubseteq, \bot, \top \rangle$

- $\langle A, \sqsubseteq \rangle$ is a $\omega$-chain complete partially $\sqsubseteq$-ordered with bottom element $\bot$ and top element $\top$
- $\langle A, \sqsupseteq \rangle$ is a $\omega$-chain complete partially $\sqsupseteq$-ordered with bottom element $\top$ and top element $\bot$

# Constraint semiring

**Field domain** : $\langle A, \sqsubseteq, \bot, \top \rangle$

- $\langle A, \sqsubseteq \rangle$ is a $\omega$-chain complete partially $\sqsubseteq$-ordered with bottom element $\bot$ and top element $\top$
- $\langle A, \sqsupseteq \rangle$ is a $\omega$-chain complete partially $\sqsupseteq$-ordered with bottom element $\top$ and top element $\bot$

**Constraint semiring** : $\langle A, +, \times, \bot, \top \rangle$ with $\sqsubseteq$ defined as $a \sqsubseteq b$ iff $a + b = b$

- $+ : A \times A \to A$ associative, commutative, idempotent **(choose)**
- $\times : A \times A \to A$ is an associative, commutative **(combine)**
- $\times$ distributes over $+$
- $\bot + a = a$, $\top + a = \top$, $\top \times a = a$, $\bot \times a = \bot$ for all $a \in A$
- $\langle A, \sqsubseteq, \bot, \top \rangle$ is a field domain of preferences

**Field domain** : $\langle A, \sqsubseteq, \bot, \top \rangle$

- $\langle A, \sqsubseteq \rangle$ is a $\omega$-chain complete partially $\sqsubseteq$-ordered with bottom element $\bot$ and top element $\top$
- $\langle A, \sqsupseteq \rangle$ is a $\omega$-chain complete partially $\sqsupseteq$-ordered with bottom element $\top$ and top element $\bot$

**Constraint semiring** : $\langle A, +, \times, \bot, \top \rangle$ with $\sqsubseteq$ defined as $a \sqsubseteq b$ iff $a + b = b$

- $+ : A \times A \to A$ associative, commutative, idempotent **(choose)**
- $\times : A \times A \to A$ is an associative, commutative **(combine)**
- $\times$ distributes over $+$
- $\bot + a = a$, $\top + a = \top$, $\top \times a = a$, $\bot \times a = \bot$ for all $a \in A$
- $\langle A, \sqsubseteq, \bot, \top \rangle$ is a field domain of preferences

**Tropical semiring** $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0 \rangle$
where the field domain is $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0 \rangle$

**Field domain** : $\langle A, \sqsubseteq, \bot, \top \rangle$

- $\langle A, \sqsubseteq \rangle$ is a $\omega$-chain complete partially $\sqsubseteq$-ordered with bottom element $\bot$ and top element $\top$
- $\langle A, \sqsupseteq \rangle$ is a $\omega$-chain complete partially $\sqsupseteq$-ordered with bottom element $\top$ and top element $\bot$

**Constraint semiring** : $\langle A, +, \times, \bot, \top \rangle$ with $\sqsubseteq$ defined as $a \sqsubseteq b$ iff $a + b = b$

- $+ : A \times A \to A$ associative, commutative, idempotent **(choose)**
- $\times : A \times A \to A$ is an associative, commutative **(combine)**
- $\times$ distributes over $+$
- $\bot + a = a$, $\top + a = \top$, $\top \times a = a$, $\bot \times a = \bot$ for all $a \in A$
- $\langle A, \sqsubseteq, \bot, \top \rangle$ is a field domain of preferences

**Tropical semiring** $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0 \rangle$
where the field domain is $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0 \rangle$

**Set semiring** $\langle 2^A, \cup, \cap, \emptyset, A \rangle$
where the field domain is $\langle 2^A, \subseteq, \emptyset, A \rangle$

# Computational fields

**Field** : tuple $\langle N, E, A, L = L_N \uplus L_E, I = I_N \uplus I_E \rangle$ where

- $N$ is a set of nodes
- $E \subseteq N \times N$ is a set of edges
- $\langle A, \sqsubseteq_A, \bot_A, \top_A \rangle$ is a field domain

  **Node evaluation** : function $N \to A$.

**Field** : tuple $\langle N, E, A, L = L_N \uplus L_E, I = I_N \uplus I_E \rangle$ where

- $N$ is a set of nodes
- $E \subseteq N \times N$ is a set of edges
- $\langle A, \sqsubseteq_A, \bot_A, \top_A \rangle$ is a field domain

  **Node evaluation** : function $N \to A$.

- $L_N$ is a set of node labels and $L_E$ is a set of edge labels
- $I_N : L_N \to (N \to A)$ defines interpretation for node labels
- $I_E : L_E \to (E \to (A \to A))$ defines interpretation for edges labels

# Computational fields

**Field** : tuple $\langle N, E, A, L = L_N \uplus L_E, I = I_N \uplus I_E \rangle$ where

- $N$ is a set of nodes
- $E \subseteq N \times N$ is a set of edges
- $\langle A, \sqsubseteq_A, \perp_A, \top_A \rangle$ is a field domain

  **Node evaluation** : function $N \to A$.

- $L_N$ is a set of node labels and $L_E$ is a set of edge labels
- $I_N : L_N \to (N \to A)$ defines interpretation for node labels
- $I_E : L_E \to (E \to (A \to A))$ defines interpretation for edges labels

**Update function** : function $(N \to A) \to (N \to A)$.

**Field domain of attribute values** : function $\langle A, \sqsubseteq_A, \perp_A, \top_A \rangle$.

**Field domain of node evaluation** : function $\langle (N \to A), \sqsubseteq_{(N \to A)}, \perp_{(N \to A)}, \top_{(N \to A)} \rangle$.

# SMuC formulas

Let $\mathcal{Z}$ be a set for Variables, $\mathcal{M}$ be a set of functions

$$\Psi ::= i \mid z \mid f(\Psi,\ldots,\Psi) \mid g \alpha \Psi \mid g \alpha \Psi \mid \mu z.\Psi \mid \nu z.\Psi$$

where:

- $i \in L_N$
- $\alpha \in L_E$
- $f \in \mathcal{M} : A^* \to A$ combines values
- $g \in \mathcal{M} : mset(A) \to A$ aggregates values
- $z \in \mathcal{Z}$ is a variable.

Given $\rho : \mathcal{Z} \to (N \to A)$, we define $[\![\cdot]\!]^F_\rho : \Psi \to (N_F \to A_F)$ as

$$
\begin{aligned}
[\![i]\!]^F_\rho &= I_F(i) \\
[\![z]\!]^F_\rho &= \rho(z) \\
[\![f(\Psi_1, \ldots, \Psi_k)]\!]^F_\rho &= \lambda n.[\![f]\!]_{A_F}([\![\Psi_1]\!]^F_\rho(n), \ldots, [\![\Psi_k]\!]^F_\rho(n)) \\
[\![g \alpha \Psi]\!]^F_\rho &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n,n')([\![\Psi]\!]^F_\rho(n')) \mid (n,n') \in E_F\ \}) \\
[\![g \alpha \Psi]\!]^F_\rho &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n',n)([\![\Psi]\!]^F_\rho(n')) \mid (n',n) \in E_F\ \}) \\
[\![\mu z.\Psi]\!]^F_\rho &= lfp\ \lambda f.[\![\Psi]\!]^F_{\rho[^f/_z]} \\
[\![\nu z.\Psi]\!]^F_\rho &= gfp\ \lambda f.[\![\Psi]\!]^F_{\rho[^f/_z]}
\end{aligned}
$$

Given $\rho : \mathcal{Z} \to (N \to A)$, we define $[\![\cdot]\!]^F_\rho : \Psi \to (N_F \to A_F)$ as

$$
\begin{aligned}
[\![i]\!]^F_\rho &= I_F(i) \\
[\![z]\!]^F_\rho &= \rho(z) \\
[\![f(\Psi_1, \ldots, \Psi_k)]\!]^F_\rho &= \lambda n.[\![f]\!]_{A_F}([\![\Psi_1]\!]^F_\rho(n), \ldots, [\![\Psi_k]\!]^F_\rho(n)) \\
[\![g \bigcirc\!\!\!\!\alpha\; \Psi]\!]^F_\rho &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n, n')([\![\Psi]\!]^F_\rho(n')) \mid (n, n') \in E_F\ \}) \\
[\![g \bullet\!\!\!\!\alpha\; \Psi]\!]^F_\rho &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n', n)([\![\Psi]\!]^F_\rho(n')) \mid (n', n) \in E_F\ \}) \\
[\![\mu z.\Psi]\!]^F_\rho &= lfp\ \lambda f.[\![\Psi]\!]^F_{\rho[^f/_z]} \\
[\![\nu z.\Psi]\!]^F_\rho &= gfp\ \lambda f.[\![\Psi]\!]^F_{\rho[^f/_z]}
\end{aligned}
$$

**Fixpoint existence** ($\lambda f.[\![\Psi]\!]^F_{\rho[^f/_z]}$ monotone and continuous)

Given $\rho : \mathcal{Z} \rightarrow (N \rightarrow A)$, we define $[\![\cdot]\!]_{\rho}^{F} : \Psi \rightarrow (N_F \rightarrow A_F)$ as

$$
\begin{aligned}
[\![i]\!]_{\rho}^{F} &= I_F(i) \\
[\![z]\!]_{\rho}^{F} &= \rho(z) \\
[\![f(\Psi_1, \ldots, \Psi_k)]\!]_{\rho}^{F} &= \lambda n.[\![f]\!]_{A_F}([\![\Psi_1]\!]_{\rho}^{F}(n), \ldots, [\![\Psi_k]\!]_{\rho}^{F}(n)) \\
[\![g \textcircled{\alpha} \Psi]\!]_{\rho}^{F} &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n, n')([\![\Psi]\!]_{\rho}^{F}(n')) \mid (n, n') \in E_F\ \}) \\
[\![g \text{\ding{108}}\!\!\alpha\ \Psi]\!]_{\rho}^{F} &= \lambda n.[\![g]\!]_{A_F}(\{\ I_F(\alpha)(n', n)([\![\Psi]\!]_{\rho}^{F}(n')) \mid (n', n) \in E_F\ \}) \\
[\![\mu z.\Psi]\!]_{\rho}^{F} &= lfp\ \lambda f.[\![\Psi]\!]_{\rho[^f/_z]}^{F} \\
[\![\nu z.\Psi]\!]_{\rho}^{F} &= gfp\ \lambda f.[\![\Psi]\!]_{\rho[^f/_z]}^{F}
\end{aligned}
$$

**Fixpoint existence** ($\lambda f.[\![\Psi]\!]_{\rho[^f/_z]}^{F}$ monotone and continuous)

**Semiring monotony:** Let $I_F$ is such that $I_F(\alpha)(e)$ is monotone for all $\alpha \in L_A$, $e \in E_A$, $\mathcal{M}$ contains only function symbols that are obtained by composing additive and multiplicative operations of the semiring. Then, every function $\lambda f.[\![\Psi]\!]_{\rho[^f/_z]}^{F}$ is monotone and continuous.

**SMuC formulas examples:**

**SMuC formulas examples:**

- Assign to every node the minimal value of the sub-graph of nodes that can reach it, for a given node label $i$
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0 \rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0 \rangle$
  - Formula: $\mu z.min(i, min \; id \; z)$

**SMuC formulas examples:**

- Assign to every node the minimal value of the sub-graph of nodes that can reach it, for a given node label $i$
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0 \rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0 \rangle$
  - Formula: $\mu z.min(i, min\ id\ z)$

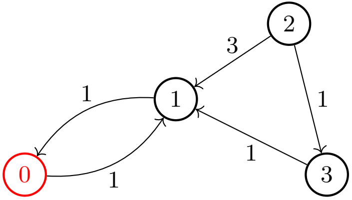
- Assign to all the nodes the union of all elements in the graph in nodes that are reachable from it, for a given node label $i$
  - Semiring: $\langle 2^A, \cup, \cap, \emptyset, A \rangle$
  - Field domain: $\langle 2^A, \subseteq, \emptyset, A \rangle$
  - Formula: $\mu z.i \cup (\cup\ id\ z)$

# SMuC formulas

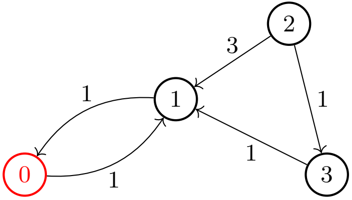
- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$

# SMuC formulas

- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1\ \alpha\ z)$
    Where $i$ is a shorthand for: $goal\ ?\ \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$



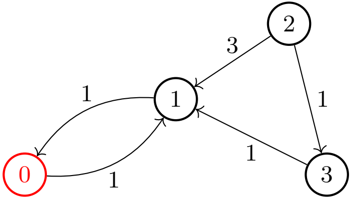
- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |

# SMuC formulas

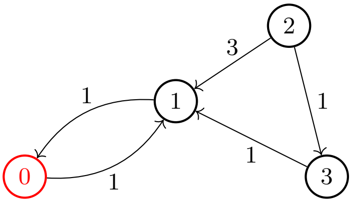
- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^1$ | $\langle 0, 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |

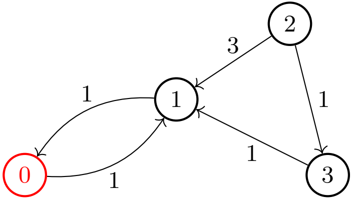
# SMuC formulas

- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^1$ | $\langle 0, 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^2$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |

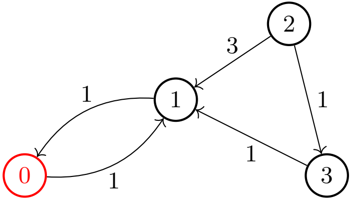
- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^1$ | $\langle 0, 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^2$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^3$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 4, 2 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |

# SMuC formulas

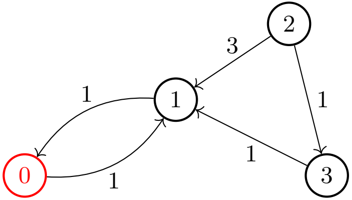
- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal \; ? \; \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^1$ | $\langle 0, 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^2$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^3$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 4, 2 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |
| $\psi^4$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 3, 2 \cdot 3 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |

# SMuC formulas

- Assign to all the nodes the minimal distance to a goal node and the path for reaching it
  - Semiring: $\langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, min, max, \bullet, \epsilon\rangle$
  - Field domain: $\langle \mathbb{R}^+ \cup \{+\infty\}, \geq, +\infty, 0\rangle \times_1 \langle N^* \cup \{\bullet\}, \sqsubseteq, \bullet, \epsilon\rangle$
  - Formula: $\mu z.min_1(i, min_1 \; \alpha \; z)$
    Where $i$ is a shorthand for: $goal\ ?\ \langle 0, self\rangle : \langle +\infty, \bullet\rangle$
  - Interpretation of node label: $I_N(self)(n) = n$
    and : $I_N(goal)(n) = true$ if $n$ is a goal node, $false$ otherwise
  - Interpretation of edge label: $I_E(\alpha)(n, n')(\langle cost, path\rangle) = \langle cost + delta, n \cdot path\rangle$




| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\psi^0$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^1$ | $\langle 0, 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^2$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle +\infty, \bullet\rangle$ | $\langle +\infty, \bullet\rangle$ |
| $\psi^3$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 4, 2 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |
| $\psi^4$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 3, 2 \cdot 3 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |
| $\psi^5$ | $\langle 0, 0\rangle$ | $\langle 1, 1 \cdot 0\rangle$ | $\langle 3, 2 \cdot 3 \cdot 1 \cdot 0\rangle$ | $\langle 2, 3 \cdot 1 \cdot 0\rangle$ |

$$P, Q ::= \texttt{skip} \mid i \leftarrow \Psi \mid P\texttt{;}Q \mid \texttt{if}\ \Psi\ \texttt{then}\ P\ \texttt{else}\ Q \mid \texttt{until}\ \Psi\ \texttt{do}\ P$$

where:

- $i \in L_N$
- $\Psi$ is a SMuC formula

# SMuC programs

$$P, Q ::= \texttt{skip} \mid i \leftarrow \Psi \mid P\texttt{;}Q \mid \texttt{if}\ \Psi\ \texttt{then}\ P\ \texttt{else}\ Q \mid \texttt{until}\ \Psi\ \texttt{do}\ P$$

where:

- $i \in L_N$
- $\Psi$ is a SMuC formula

**State of computation:** pair of program and field

**Memory stores:** is represented by interpretation function of the field

**Semantics:** given via transition system $\rightarrow \subseteq (P \times \mathcal{F})^2$

# Emergent to local behavior

- **Simple assignment program**: no fixpoints
- **Asynchronous agreement**: we use a tree-based infrastructure that spans the complete field
- **Distributed field infrastructure**: $\langle$ Field , spanning Tree $\rangle$
- **Distributed fragment:** $d = n[\ S \mid \iota : \chi : k\ ]$
  - $n \in N$ is a node
  - $S$ is the simple assignment program currently executed by $n$
  - $\iota : N \to L_N \to A$ is a partial interpretation of node labels at $n$
  - $\chi$ is an agreement store
  - $k$ is an agreement counter

# Emergent to local behavior

- **Simple assignment program**: no fixpoints
- **Asynchronous agreement**: we use a tree-based infrastructure that spans the complete field
- **Distributed field infrastructure**: $\langle$ Field , spanning Tree $\rangle$
- **Distributed fragment:** $d = n[\ S \mid \iota : \chi : k\ ]$
  - $n \in N$ is a node
  - $S$ is the simple assignment program currently executed by $n$
  - $\iota : N \to L_N \to A$ is a partial interpretation of node labels at $n$
  - $\chi$ is an agreement store
  - $k$ is an agreement counter
- **Evolution of fragments:** $d_n \to^{msg} d_n$
- **Evolution of distributed execution:** $D \Rightarrow^{msg} D$ where $D = \{d_n\}_{n \in N}$

# Robustness against node unavaliability

When we compute the fixpoint of an update function $\psi : (N \to A) \to (N \to A)$

- nodes are allowed to proceed at different speeds
- nodes may remain inactive for some iterations
- nodes don't wait for communications, they use caching

# Robustness against node unavaliability

When we compute the fixpoint of an update function $\psi : (N \to A) \to (N \to A)$

- nodes are allowed to proceed at different speeds
- nodes may remain inactive for some iterations
- nodes don't wait for communications, they use caching

**We guarantee** that the same fixpoint of a synchronous execution is reached, under reasonable conditions:

- we have only a finite number of nodes
- every node execute infinitely often (fair strategy)
- the update function $\psi$ is monotone (remember semiring operations)

When we compute the fixpoint of an update function $\psi : (N \to A) \to (N \to A)$

- nodes are allowed to proceed at different speeds
- nodes may remain inactive for some iterations
- nodes don't wait for communications, they use caching

**We guarantee** that the same fixpoint of a synchronous execution is reached, under reasonable conditions:

- we have only a finite number of nodes
- every node execute infinitely often (fair strategy)
- the update function $\psi$ is monotone (remember semiring operations)

**Actually** to guarantee that some fixpoint is reached at all we need also to add this condition

- $A$ has finite partially ordered chains **only**

# Robustness against node failures

We consider the possibility for a node to fail, it may stay inactive for a while and then resumes and **enters a backup state** it had in a previous iteration (e.g. the initial one)

# Robustness against node failures

We consider the possibility for a node to fail, it may stay inactive for a while and then resumes and **enters a backup state** it had in a previous iteration (e.g. the initial one)

**We guarantee** that the same fixpoint of a synchronous no-failure execution is reached, under reasonable conditions:

- we have only a finite number of nodes
- every node execute infinitely often (fair strategy)
- the update function $\psi$ is monotone (remember semiring operations)
- **at some point the system enters a condition in where no more failures occur**

# CFC

# Computational Field Calculus main ideas

**Fields:** networks with attributes on nodes.

**Execution:** only a simple generic construct for iteration.

**Values:** from a sound type system

- number and boolean as elementary data types
- we can build pairs
- we can build filed types

Conditional construct `if` allows the restriction to a **sub-region of the network**

| | |
|---|---|
| $\mathtt{e} ::= \mathtt{x} \mid \mathtt{l} \mid (\mathtt{o}\bar{\mathtt{e}}) \mid (\mathtt{d}\bar{\mathtt{e}}) \mid (\mathtt{rep\ x\ w\ e}) \mid (\mathtt{nbr\ e}) \mid (\mathtt{if\ e\ e\ e})$ | *expression* |
| $\mathtt{l} ::= \mathtt{n} \mid \mathtt{b} \mid \langle \mathtt{l}, \mathtt{l} \rangle$ | *lacal value* |
| $\mathtt{w} ::= \mathtt{x} \mid \mathtt{l}$ | *variable or lacal value* |
| $\mathtt{D} ::= (\mathtt{def\ d}(\bar{\mathtt{x}})\mathtt{e})$ | *user-defined function declaration* |
| $\mathtt{P} ::= \bar{\mathtt{D}}\mathtt{e}$ | *program* |

Where:

- $\mathtt{n}$ is a number
- $\mathtt{b}$ is a boolean
- $\mathtt{x}$ is a variable name
- $\mathtt{o}$ is a builtin function name
- $\mathtt{d}$ is a user-defined function name

# CFC semantics

$$\mathtt{e} ::= \mathtt{x} \mid \mathtt{l} \mid (\mathtt{o}\bar{\mathtt{e}}) \mid (\mathtt{d}\bar{\mathtt{e}}) \mid (\mathtt{rep\ x\ w\ e}) \mid (\mathtt{nbr\ e}) \mid (\mathtt{if\ e\ e}'\ \mathtt{e}'') \qquad \textit{expression}$$

Informal semantics:

- $(\mathtt{o}\bar{\mathtt{e}})$ is the composition between the fields $\bar{\mathtt{e}} = (\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_n)$
- $(\mathtt{d}\bar{\mathtt{e}})$ is the application of the function $\mathtt{d}$ to the fields $\bar{\mathtt{e}} = (\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_n)$
- $(\mathtt{rep\ x\ w\ e})$ is the field obtained by starting from configuration $\mathtt{w}$ and updating through time using $\mathtt{e}$ as update function where $\mathtt{x}$ represent the actual state of the field
- $(\mathtt{nbr\ e})$ is the propagation of $\mathtt{e}$ to neighboring nodes (field values)
- $(\mathtt{if\ e\ e}'\ \mathtt{e}'')$ is the field $\mathtt{e}'$ in nodes where $\mathtt{e}$ evaluates to `true` and $\mathtt{e}''$ in nodes where $\mathtt{e}$ evaluates to `false`

# CFC examples

- Field that assigns to all the nodes the minimum reachable value in a given field

```
( def gossip-min ( source )
    ( rep d source ( min-hood ( nbr d ))))
```

# CFC examples

- Field that assigns to all the nodes the minimum reachable value in a given field

```
( def gossip-min ( source )
    ( rep d source ( min-hood ( nbr d ))))
```

- Field that assigns to each node the minimal distance to a source node

```
( def distance-to ( source )
     ( rep d infinity
         ( mux source 0 ( min-hood ( +[f,f] ( nbr d ) (nbr-range)))))))
```

# CFC examples

- Field that assigns to all the nodes the minimum reachable value in a given field

```
( def gossip-min ( source )
    ( rep d source ( min-hood ( nbr d ))))
```

- Field that assigns to each node the minimal distance to a source node

```
( def distance-to ( source )
     ( rep d infinity
         ( mux source 0 ( min-hood ( +[f,f] ( nbr d ) (nbr-range)))))))
```

- Field that assigns to each node the minimal distance to a source node avoiding obstacle nodes

```
( def distance-obs-to ( source, obstacle )
     ( if ( not obstacle ) ( distance-to source ) infinity )))
```

**Execution is partially synchronous,** in each round a device:

- sleeps for some limited time
- wakes up
- gathers information about messages received while asleep
- performs his field evaluation
- emits a message to all neighbours

## CFC semantiscs

**Execution is partially synchronous,** in each round a device:

- sleeps for some limited time
- wakes up
- gathers information about messages received while asleep
- performs his field evaluation
- emits a message to all neighbours

**Operational semantics:** based on runtime expression syntax

- **Annotations:** for transient partial run-time information about the computation
- **Superscript:** for durable partial run-time information about the computation

*Congruence* and *alignment* context are used to impose an order of evaluation to subexpressions

A type system to guarantee well-formedness of expressions

# Conclusion

# SMuC vs CFC

**Soft Mu-Calculus for Computational Fields (SMuC)**

- based on **fixpoints** computation
- composition of the values from neighboring nodes expressed by **mu-calculus-like formulas**
- domains for values built from **constraint semiring**
- **robust against node unavailability and failure**
- **synchronization-based** constructs `if` and `until`

**Computational Fields Calculus (CFC)**

- minimal
- it allows the restriction of a field computation to a **sub-region** of the network
- **simpler** approach

## `rep` vs $\mu$ and $\nu$

`rep` in CFC

- based on the idea that the computation never ends
- is more handy (you specify the beginning state and the update rule)

# `rep` vs $\mu$ and $\nu$

`rep` in CFC

- based on the idea that the computation never ends
- is more handy (you specify the beginning state and the update rule)

$\mu$ and $\nu$ in SMuC

- based on the idea that the computation must end
- you have to think domain-based
- automatic guarantee of termination
- requires you more, gives you more

# `rep` vs $\mu$ and $\nu$

`rep` in CFC

- based on the idea that the computation never ends
- is more handy (you specify the beginning state and the update rule)

$\mu$ and $\nu$ in SMuC

- based on the idea that the computation must end
- you have to think domain-based
- automatic guarantee of termination
- requires you more, gives you more

(`rep` in CFC and `until` in SMuC are quite the same)

# Two very different flavour of `if` construct

**`if` from CFC and `if` from SMuC only share the name**

`if` from CFC makes two separate computations

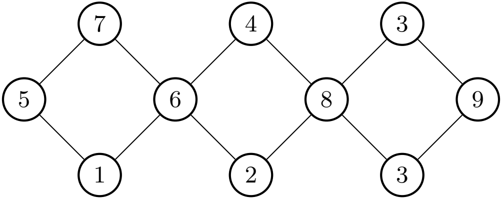
- one for the sub-network that evaluates the guard to true
- one for the sub-network that evaluates the guard to false

`if` from SMuC is synchronization based

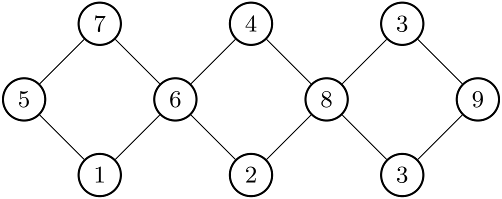
- if all the network evaluates the guard to true execute the first branch
- if at least one node evaluates the guard to false execute the second branch

# Two very different flavour of `if` construct

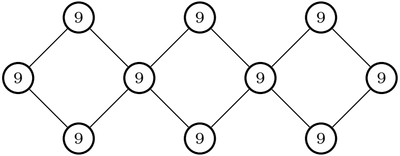
**Example:** `if even then cmp-min else cmp-max`



**Example:** `if even then cmp-min else cmp-max`

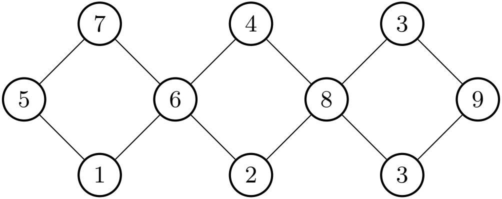



**In SMuC**

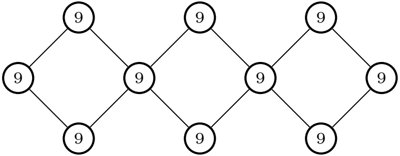


# Two very different flavour of if construct

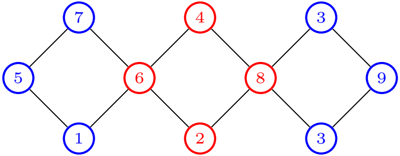
**Example:** `if even then cmp-min else cmp-max`




**In SMuC**

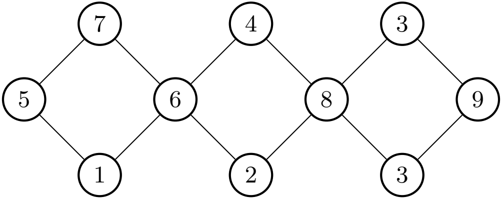



**In CFC**

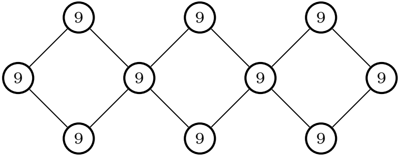


# Two very different flavour of `if` construct

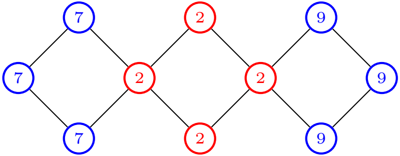
**Example:** `if even then cmp-min else cmp-max`




**In SMuC**




**In CFC**



**Can we simulate the two constructs ?**

**Can we simulate the two constructs ?**

- SMuC `if` inside CFC

  We need to create support for distributed synchronization
  (i.e. write code for what in SMuC is runtime support)

# Two very different flavour of `if` construct

**Can we simulate the two constructs ?**

- SMuC `if` inside CFC

  We need to create support for distributed synchronization
  (i.e. write code for what in SMuC is runtime support)

- CFC `if` inside SMuC

  Problem for semantic compositionality (e.g. `if` $\Psi_0$ `then` $\Psi_1$ `else` $\Psi_1$ )
  We can obtain the same behavior only with ad hoc rewriting

# References

# References


- *Jacob Beal and Mirko Viroli*. **Space–time programming**. Phil.Trans.R.Soc. A 373: 20140220, 2015.
- *Alberto Lluch-Lafuente, Michele Loreti, and Ugo Montanari*. **A fixpoint-based calculus for graph-shaped computational fields**. In Tom Holvoet and Mirko Viroli, editors, Coordination Models and Languages - 17th IFIP WG 6.1 International Conference, COORDINATION 2015, Held as Part of the 10th International Federated Conference on Distributed Computing Techniques, DisCoTec 2015, Grenoble, France, June 2-4, 2015, Proceedings, volume 9037 of Lecture Notes in Computer Science, pages 101–116. Springer, 2015.
- *Lluch Lafuente Alberto, Loreti Michele, Montanari Ugo*. **Asynchronous Distributed Execution Of Fixpoint-Based Computational Fields**. Logical Methods in Computer Science, Volume 13, Issue 1 (March 22, 2017) lmcs:3212
- *J. Beal, J. Bachrach*, **Infrastructure for engineered emergence in sensor/actuator networks**, IEEE Intell. Syst. 21 (March/April 2006) 10–19.
- *Ferruccio Damiani, Mirko Viroli and Jacob Beal*. **A type-sound calculus of computational fields**. Science of Computer Programming, 117:17 – 44, 2016.