Diagnosis with Dynamic MSCs

(Ongoing work)

Benedikt Bollig\textsuperscript{1}, Stefan Haar\textsuperscript{2}, Loïc Hélouët\textsuperscript{3}

\textsuperscript{1}LSV/CNRS, \textsuperscript{2}LSV/INRIA, \textsuperscript{3}IRISA/INRIA
Motivations

Observation = code instrumentation, traffic sniffing & filtering, ...
Diagnosis = Fault detection, explanation retrieval, ...
From a model $H$ and an observation $O$, find all explanations for $O$ in $H$.

Unfolding: synchronize execution of the observation with all compatible runs of the model (may not terminate)
From a model $H$ and an observation $O$, find all explanations for $O$ in $H$.

**Generator:** build a new model in which all behaviors embed the observation (not always feasible)
Motivations

Done for:

- Automata (static, fault detection) [Sampath & al]
- Petri nets (static archi., unfolding) [Benveniste & al, Chatain & al]
- High-level MSCs (static archi., generator) [Gazagnaire & al]
- Graph grammars (dynamic archi., unfolding) [Baldan & al]
Motivations

Diagnosis for:

- Partial order model (can avoid costly interleavings)
- with dynamic aspects + buffering
- Compute a generator
- Compositionality issues
Motivations

Diagram:

- **P1**
- **P2**
- **Observer(s)**
- **spawn(P8)**
- **P3**
- **P4**
- **P5**
- **P8**
- **P6**
- **P7**

Nodes and Edges:

- **Im**
- **e**
- **?n**
- **Observation (partial order)**
- **Model (MSC grammar)**
- **Diagnoser**
- **Explanations (MSC grammar ?)**

Diagnosis with Dynamic MSCs – p. 7/43
Outline

- Message Sequence Charts, Observations
- MSC Grammars
- MSO for MSCs
- Diagnosis
- Conclusions
Message Sequence Charts

\[ M = (E, \leq, \alpha, \varphi, \mu) \in \mathcal{M} : \]

- \( \alpha \subseteq E \times \Sigma \): labeling,
- \( \varphi \subseteq E \times \mathbb{N} \): locality,
- \( \mu \subseteq E \times E \): messages

\[ x \leq y \text{ iff} \]

- sequential ordering on a process, or
- causal chain from \( x \) to \( y \)
$x \leq_{1o2} y$ iff

- $x \leq_1 y$, or
- $x \leq_2 y$, or
- $x \leq_1 z$, $z' \leq_2 y$ and $z$, $z'$ on the same process
Observations

- Choose an observation alphabet
  \[ \Sigma_{obs} \]

- Every observed process
  - reports occurrences of actions in \[ \Sigma_{obs} \]
  - can maintain tags to retrieve causality (e.g., vectorial clocks)

- Observer receives collected events and builds
  \[ O = (E_O, \leq_O, \alpha_O, \varphi_O, \mu_O) \]
Find an embedding \( h : E_O \rightarrow E_M \) that is compatible:

- with labeling: \( \alpha(x) = \alpha(h(x)) \)
- with ordering: \( x \leq_O y \implies h(x) \leq_M h(y) \)
- with locality of events: 
  \[ \varphi(x) = \varphi(y) \iff \varphi(h(x)) = \varphi(h(y)) \]

...
High-level MSCs (HMSCs)

- Partial order model, infinite, non-regular behaviors
- Diagnosis with generator works [HelouetWodes06]

But: *Finite* set of processes only
The important in MSC grammars is the shape of a scenario, not the identity of processes.

Named MSCs: $\mathcal{M}$

Rely on process identifiers: $\pi_1, \ldots, \pi_k$.

$$(M_1, \nu_1) = \begin{array}{c}
\pi_1 \\
\text{start} \\
\text{spawn} \\
\pi_2 \\
\text{start}
\end{array}$$
Named PMSCs: $\mathcal{M}$

\[ (M_2, \mu_2) = \]

- Process IDs passed from one process to another
- The name of non-identified processes is meaningless
- Glue a behavior on identified processes
MSC Grammars

Concatenation: \((M_1, \nu_1) \circ (M_2, \mu_2)\)
MSC Grammars

Concatenation: \((M_1, \nu_1) \circ (M_2, \mu_2)\)
MSC Grammars

In short: context free grammar with named MSCs and PMSC as terminals

- Subset of Dynamic MSCs [Leucker & al 02]
- A kind of graph grammar...
MSC Grammars

Derivations:
... as usual but writing PMSCs instead of words/hypergraphs

Axiom

```
1
  a
Π₁ Π₂
A
```
MSC Grammars

Derivations:
... as usual but writing MSCs instead of words/hypergraphs

1

a

Spawn

Π₁

B

Π₂

m

2
MSC Grammars

Derivations:
... as usual but writing MSCs instead of words/hypergraphs
MSC Grammars

Derivations:
... as usual but writing MSCs instead of words/hypergraphs

```
1 -- Spawn -- 2
    | a     |     |
    |       | Spawn|
    | b     |     |
    |       |     |
3 -- Spawn -- 4
    | b     |     |
    |       |     |
    | m     | n   |
    |       |     |
  m
```
MSC Grammars

Parse Tree

Axiom

1

\[ \Pi_1 \Pi_2 \]

A

\[ M_1 \]

Rule A

\[ \Pi_1 \Pi_2 \]

1

Spawn

2

\[ M_2 \]

\[ M_3 \]

Axiom

\[ M_1 \]

\[ M_2 \]

A

B

\[ M_3 \]

M_4

B

M_5

M_6

Diagnosis with Dynamic MSCs – p. 23/43
Language of an MSC grammar $G$:

$$L(G) = \{ M \in \mathbb{M} \mid \nu_G \xrightarrow{\ast}_G (M', \nu) \text{ for some } M' \cong M \text{ and } \nu \}.$$  

All the MSCs that can be derived from the axiom of $G$.

Nb: $L(G)$ does not differentiate between isomorphic MSCs.
Tree automata

\[ TA = (Q, Q_F, \mathcal{F}, \delta) \]

- \( Q \): set of states
- \( Q_F \subseteq Q \): set of final states
- \( \mathcal{F} \): symbols (terminal and non terminal)
- \( \delta \subseteq \mathcal{F} \times \bigcup_{1..K} Q^i \times Q \): transition relation

\[ f(q_1(x_1), \ldots q_n(x_n)) \rightarrow q(f(x_1), \ldots f(x_n)) \]

A run of a TA on a tree \( T = (N, \rightarrow) \) is a mapping \( r : N \rightarrow Q \). \( r \) is successful if \( r(\text{root}) \in Q_F \)

TAs are recognizers for regular tree languages.
The derivation trees of a context free grammar are regular tree languages. The converse does not always hold, except for **local tree languages**.
Tree automata

\[ T_A = \bigcup M_i \]

\begin{align*}
\text{Axiom} & (q_{ax}(x), q_A(y)) \\
A(q_2(x), q_B(y), q_3(z)) & \\
B(q_4(x), q_B(y), q_5(z)) & \\
B(q_6(x)) & \\
q_1 M_1 & \\
q_2 M_2 & \\
q_4 M_4 & \\
q_5 M_5 & \\
q_6 M_6 & \\
A q_A & \\
B q_B & \\
q_B' & \\
\end{align*}

\( G \iff T_A_G(local) \)
MSO over MSCs [Leucker & al02]

\[ \varphi ::= \text{lab}_a(x) \]

- \( x \) is an event labelled by a

\[ (u, x) \rightarrow (v, y) \]

- \( (x, y) \) is a message from \( u \) to \( v \)

\[ x \preceq y \]

- \( x \) is the immediate predecessor of \( y \) on some process

\[ x \in X \quad u \in U \]

\[ \neg \varphi \quad \varphi_1 \land \varphi_2 \]

- event/event set quantifier

\[ \exists x \varphi \quad \exists X \varphi \]

\[ \exists u \varphi \quad \exists U \varphi \]

- process/process set quantifier
**Theorem 1**  Let $O$ be an observation over a set of events $e_1 \ldots e_n$, and $M$ be a MSC and $\Sigma_{\text{Obs}}$ be the observation alphabet. Then $O \triangleright_{\Sigma_{\text{obs}}} M$ if and only if $M \models \varphi_O$, where $\varphi_O$ is the formula

\[
\exists x_1, \ldots x_n, \quad \bigwedge_{x \leq O y} \text{causalChain}(x, y) \\
\quad \wedge \bigwedge_{i \in 1..n} \text{lab}(x_i) = \lambda_O(e_i) \\
\quad \wedge \bigwedge_{i \in 1..n} \nexists z, \text{LocalPredecessor}(x_i, z) \wedge \text{lab}(z) \in \Sigma_{\text{Obs}} \\
\quad \wedge z \notin x_1, \ldots x_n
\]
\( \varphi_O \ ::= \ \exists x, y, z, t, u, v, w, \)
\[
lab_a(x) \land lab_b(y) \land lab_b(z) \land lab?_n(t) \\
\land LocalPredecessor(x, t) \\
\land (u, x) \leq (v, y) \land (u, x) \leq (w, z)
\]
Results

Theorem 2  \textit{MSO over MSCs is decidable for MSC grammars}

Corollary 1  \textit{Diagnosis with MSC grammars is decidable}
Interpreted tree automata that recognise parse trees

- guess $\gamma : V_{\varphi_O} \rightarrow E_M \cup P_M$
- infer $\pi_i \leq x$, $x \leq \pi_i'$, ...
- infer $\pi_i \leq \pi_i'$
- infer atoms of $\varphi_O$ that hold at $M$

$\exists x, y, z, u, \text{lab}_a(x), \text{lab}_b(y), \text{lab}_b(z)$

$\pi_1 \leq x, y, z \land \pi_2 \leq y, z \land x \leq \pi_1', \pi_2' \land z \leq \pi_1'$

$\pi_1 \leq \pi_1' \land \pi_1 \leq \pi_2' \land \pi_2 \leq \pi_1' \land \pi_2 \leq \pi_2'$
Proof sketch

Accepting states: \( q_{ax} \times \varphi \) such that \( \varphi \implies \varphi_O \)
Parse trees are decorated with:

- Interpretations over $V_{\varphi_O}$ on leaves (MSCs) (finite)
- Predicates denoting causal chains in a subtree (finite, involves identifiers or chosen processes in the subtree)
- Communication structure (mobility of processes identifiers)
- Sub-formulae of $\varphi_O$ that hold in the subtree

Tree automata transitions depend on consistency of labelings
We obtain a generator for all explanations of $O$
Conclusion

We have:

- Dynamic scenario model
- MSO/diagnosis decidable for it
- Generator comes for free as a consequence of decorated parse tree
- Compositionnality comes for free as a consequence of embeddings properties (NB: \( \neq \) obs., same grammar)
Future Work

Not a surprising result:

- MSC Grammars are graph grammars: all results apply [Courcelle]
- also subset of Dynamic MSCs [Leucker]
- Decidability only: MSO usually means exponential blowup!
Future Work

... that gives clues for efficient algorithms

- MSC Grammars as Hyperedge replacement + activation rule
- Not any MSO formula, not any kind of graph:
  - use the information on the model to avoid useless transitions in the TA
  - formula \( \approx \) looking a sequence on each process, plus some inter process ordering?
On compositionality

Diagnosis with Dynamic MSCs – p. 39/43
Definition 1  A (dynamic) MSC grammar is a quadruple $G = (\Pi, \mathcal{N}, S, \rightarrow)$ where

- $\Pi$ and $\mathcal{N}$ and are nonempty finite sets of process identifiers and non-terminals, respectively,
- $S \in \mathcal{N}$ is the start non-terminal, and
- $\rightarrow$ is a finite set of rules
MSC Grammars
(Formal defs)

Definition 2  A rule is a triple \( r = (A, \alpha, f) \) with

- \( A \in \mathcal{N} \) non-terminal,
- \( \alpha \) expression over \( \mathcal{N} \) and \( \Pi \)
- \( f : \text{Free}(\alpha) \rightarrow \Pi \) associates process identifiers to free processes of \( \alpha \).

We may write \( A \xrightarrow{f} \alpha \).
MSO over MSCs

Causal chain : folklore [Madhusudan&al05]

\[
\text{causalChain}(x, y) ::= \exists X, x \in X, y \in X, \\
\forall Y \subset X, \exists x' \in \max(X \setminus Y), \\
\text{closed}(Y) \implies \exists y' \in \min(Y), \\
x' < y' \lor x \rightarrow y
\]

\[
\text{closed}(Y) ::= \nexists x, \nexists y \in Y, (x < y \lor x \rightarrow y) \land x \notin Y
\]
MSC grammars

MSC Grammars vs Dynamic MSCs [Leucker & al02]

- Dyn. MSCs more expressive than MSC grammars
- $L(G)$ has to be evaluated recursively (LR in MSC grammars)
- Implementation model for MSC grammars

Questions on MSC grammars

- $L(G) = \emptyset$?
- Realizability
- Implementation