Our basic data structure

A mailbox tree on a finite set of natural number I (m-tree on I for short) is a tree $T = (N, E)$ where:

- $N \subseteq 2^I$ is the set of nodes and $\emptyset \in N$ is the root the tree
- the cardinality of each node equals its level in T
- for any siblings X and Y, $X \cap Y = \emptyset$ and X precedes Y if $\max X \leq \max Y$
- for each edge $(X, Y) \in E$, $X \subset Y$ and $\max Y \notin X$.

Our basic data structure

A mailbox tree on a finite set of natural number I (m-tree on I for short) is a tree $T = (N, E)$ where:

- $N \subseteq 2^I$ is the set of nodes and $\emptyset \in N$ is the root the tree
- the cardinality of each node equals its level in T
- for any siblings X and Y, $X \cap Y = \emptyset$ and X precedes Y if $\max X \leq \max Y$
- for each edge $(X, Y) \in E$, $X \subset Y$ and $\max Y \notin X$.

An m-tree T on I is consistent when, for each level $h > 0$ of T,
$\bigcup \{X \in T \mid X \text{ is at level } h\} = I$

Growing trees

Given a natural number i and a tree $T = (N, E)$ s.t. the elements of N are subsets of numbers not including i

Growing trees

Given a natural number i and a tree $T = (N, E)$ s.t. the elements of N are subsets of numbers not including i

The ramification of T with i is the tree $\mathrm{r}(T, i) = (N', E')$ s.t.

$$N' = N \cup \{X \cup \{i\} \mid X \in N\} \qquad \text{and} \qquad E' = E \cup \{(Y \setminus \{\max Y\}, Y) \mid Y \in N' \setminus \{\emptyset\}\}$$

Growing trees

Given a natural number i and a tree $T = (N, E)$ s.t. the elements of N are subsets of numbers not including i

The ramification of T with i is the tree $\mathrm{r}(T, i) = (N', E')$ s.t.

$$N' = N \cup \{X \cup \{i\} \mid X \in N\} \qquad \text{and} \qquad E' = E \cup \{(Y \setminus \{\max Y\}, Y) \mid Y \in N' \setminus \{\emptyset\}\}$$

Proposition

Ramification is a commutative internal operation on m-trees.

Assignments

Fix a mailbox $\mathcal{M}$ and a join pattern J if γ with $J = \mu_1 \wedge \ldots \wedge \mu_p$

Assignments

Fix a mailbox $\mathcal{M}$ and a join pattern J if γ with $J = \mu_1 \wedge \ldots \wedge \mu_p$

Let $c : \{1, \ldots, p\} \to 2^{\{1,\ldots,|\mathcal{M}|\}}$ be

$$c(i) = \left\{ 1 \leq j \leq |\mathcal{M}| \;\middle|\; \text{there is a substitution } \sigma \text{ such that } \mu_i\sigma = \mathcal{M}[j] \right\} \quad (2)$$

Assignments

Fix a mailbox $\mathcal{M}$ and a join pattern J if γ with $J = \mu_1 \wedge \ldots \wedge \mu_p$

Let $c : \{1, \ldots, p\} \to 2^{\{1,\ldots,|\mathcal{M}|\}}$ be

$$c(i) = \left\{ 1 \leq j \leq |\mathcal{M}| \;\middle|\; \text{there is a substitution } \sigma \text{ such that } \mu_i\sigma = \mathcal{M}[j] \right\} \quad (2)$$

Let $\mathrm{asgn}(\mathcal{M}, J)$ be the set of $\mathcal{M}$-assignments for J, that is injective maps $a : \{1, \ldots, p\} \to \{1, \ldots, |\mathcal{M}|\}$ such that $c(i) \neq \emptyset$ for all $1 \leq i \leq p$.

Assignments

Fix a mailbox $\mathcal{M}$ and a join pattern J if γ with $J = \mu_1 \wedge \ldots \wedge \mu_p$

Let $\mathsf{c} : \{1, \ldots, p\} \to 2^{\{1,\ldots,|\mathcal{M}|\}}$ be

$$\mathsf{c}(i) \quad = \quad \left\{ 1 \leq j \leq |\mathcal{M}| \;\middle|\; \text{there is a substitution } \sigma \text{ such that } \mu_i\sigma = \mathcal{M}[j] \right\} \qquad (2)$$

Let $\mathrm{asgn}(\mathcal{M}, J)$ be the set of $\mathcal{M}$-assignments for J, that is injective maps $\mathsf{a} : \{1, \ldots, p\} \to \{1, \ldots, |\mathcal{M}|\}$ such that $\mathsf{c}(i) \neq \emptyset$ for all $1 \leq i \leq p$.

An assignment $\mathsf{a} \in \mathrm{asgn}(\mathcal{M}, J)$ is valid for the guard γ if $\gamma\sigma_{\mathsf{a}}$ is true
(with σ_{a} a choice of substitutions induced by c)

From m-trees to assignments

The assignment tree of a join pattern J if γ w.r.t. mailbox $\mathcal{M}$ $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ is the pair $(T, \mathfrak{a})$ where, letting $I = \mathsf{c}(\{1, \ldots, p\})$ with c as in (2),

- $T = (N, E)$ is the subtree up-to level p of $\mathsf{r}((\{\emptyset\}, \emptyset), I)$ and
- the map of candidate assignments $\mathfrak{a} : N \to 2^{\mathsf{asgn}(\mathcal{M}, J)}$ is such that

$$\mathfrak{a}(X) \;=\; \{\mathsf{a} \in \mathsf{asgn}(\mathcal{M}, J) \;\big|\; \mathsf{a} \text{ is valid for } \gamma \text{ and } \mathsf{cod}\,\mathsf{a} = X\}$$

for each node $X \in N$,

Then, J if γ is resolved in $\mathcal{M}$ if there is a leaf X in $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ such that $\mathfrak{a}(X) \neq \emptyset$.

Theorem

For all $\mathsf{a} \in \mathsf{asgn}(\mathcal{M}, J)$, $\mathsf{cod}\,\mathsf{a} \in \mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$.

Proof.

By induction on p using the definition of ramification. □

Order, order!

An indexing sequence is a non-empty sequence $\mathcal{I}$ of pairwise-distinct and strictly positive natural numbers

Order, order!

An indexing sequence is a non-empty sequence $\mathcal{I}$ of pairwise-distinct and strictly positive natural numbers

Given a sequence $\mathcal{S}$ and an indexing sequence $\mathcal{I} = i_1 \cdot \ldots \cdot i_n$ such that $1 \leq i_h \leq |\mathcal{S}|$ for each h, the $\mathcal{I}$-slice of $\mathcal{S}$ is the sequence $\mathcal{S}[\mathcal{I}] = \mathcal{S}[i_1] \cdot \ldots \cdot \mathcal{S}[i_n]$

Order, order!

An indexing sequence is a non-empty sequence $\mathcal{I}$ of pairwise-distinct and strictly positive natural numbers

Given a sequence $\mathcal{S}$ and an indexing sequence $\mathcal{I} = i_1 \cdot \ldots \cdot i_n$ such that $1 \leq i_h \leq |\mathcal{S}|$ for each h, the $\mathcal{I}$-slice of $\mathcal{S}$ is the sequence $\mathcal{S}[\mathcal{I}] = \mathcal{S}[i_1] \cdot \ldots \cdot \mathcal{S}[i_n]$

The lexicographic order of a set $\mathbb{S}$ totally ordered by $\sqsubseteq$, is the relation $\leq_{\text{lex}}$ on sequences in $\mathbb{S}^*$ induced by the rules inductively defined as:

$$\frac{}{\epsilon \leq_{\text{lex}} \epsilon} \qquad \frac{\mathcal{S} \leq_{\text{lex}} \mathcal{S}'}{s \cdot \mathcal{S} \leq_{\text{lex}} s \cdot \mathcal{S}'} \qquad \frac{s \sqsubseteq s' \quad s \neq s'}{s \cdot \mathcal{S} \leq_{\text{lex}} s' \cdot \mathcal{S}'}$$

Order, order!

An indexing sequence is a non-empty sequence $\mathcal{I}$ of pairwise-distinct and strictly positive natural numbers

Given a sequence $\mathcal{S}$ and an indexing sequence $\mathcal{I} = i_1 \cdot \ldots \cdot i_n$ such that $1 \leq i_h \leq |\mathcal{S}|$ for each h, the $\mathcal{I}$-slice of $\mathcal{S}$ is the sequence $\mathcal{S}[\mathcal{I}] = \mathcal{S}[i_1] \cdot \ldots \cdot \mathcal{S}[i_n]$

The lexicographic order of a set $\mathbb{S}$ totally ordered by $\sqsubseteq$, is the relation $\leq_{\text{lex}}$ on sequences in $\mathbb{S}^*$ induced by the rules inductively defined as:

$$\frac{}{\epsilon \leq_{\text{lex}} \epsilon} \qquad \frac{\mathcal{S} \leq_{\text{lex}} \mathcal{S}'}{s \cdot \mathcal{S} \leq_{\text{lex}} s \cdot \mathcal{S}'} \qquad \frac{s \sqsubseteq s' \quad s \neq s'}{s \cdot \mathcal{S} \leq_{\text{lex}} s' \cdot \mathcal{S}'}$$

Note: $\mathcal{S} \leq_{\text{lex}} \mathcal{S}'$ ***implies*** $|\mathcal{S}| = |\mathcal{S}'|$

Order, order!

An indexing sequence is a non-empty sequence $\mathcal{I}$ of pairwise-distinct and strictly positive natural numbers

Given a sequence $\mathcal{S}$ and an indexing sequence $\mathcal{I} = i_1 \cdot \ldots \cdot i_n$ such that $1 \leq i_h \leq |\mathcal{S}|$ for each h, the $\mathcal{I}$-slice of $\mathcal{S}$ is the sequence $\mathcal{S}[\mathcal{I}] = \mathcal{S}[i_1] \cdot \ldots \cdot \mathcal{S}[i_n]$

The lexicographic order of a set $\mathbb{S}$ totally ordered by $\sqsubseteq$, is the relation $\leq_{\text{lex}}$ on sequences in $\mathbb{S}^*$ induced by the rules inductively defined as:

$$\frac{}{\epsilon \leq_{\text{lex}} \epsilon} \qquad \frac{\mathcal{S} \leq_{\text{lex}} \mathcal{S}'}{s \cdot \mathcal{S} \leq_{\text{lex}} s \cdot \mathcal{S}'} \qquad \frac{s \sqsubseteq s' \quad s \neq s'}{s \cdot \mathcal{S} \leq_{\text{lex}} s' \cdot \mathcal{S}'}$$

Note: $\mathcal{S} \leq_{\text{lex}} \mathcal{S}'$ ***implies*** $|\mathcal{S}| = |\mathcal{S}'|$

The length-biased lexicographic order $\leqslant_{\text{slex}}$ is the relation induced by

$$\frac{n = |S'| \leq |S| \qquad sort_{\sqsubseteq}(\mathcal{S})[1 \cdot \ldots \cdot n] \leqslant_{\text{lex}} sort_{\sqsubseteq}(\mathcal{S}')}{\mathcal{S} \leqslant_{\text{slex}} \mathcal{S}'}$$

Fair resolution

Given $\mathcal{M}$ and $J = \mu_1 \wedge \ldots \wedge \mu_p$, let $\preceq$ be the total order on $\mathrm{asgn}(\mathcal{M}, J)$ defined by

$$\mathsf{a} \preceq \mathsf{a}' \quad \text{if} \quad \langle \mathsf{a}(1) \cdot \ldots \cdot \mathsf{a}(p) \rangle \leq_{\mathsf{lex}} \langle \mathsf{a}'(1) \cdot \ldots \cdot \mathsf{a}'(p) \rangle$$

The fair resolution of $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ is the minimal assignment in $\mathfrak{a}(X)$ wrt $\preceq$ where X is first node in a depth-first visit of the assignment tree $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ at level p whose candidate assignment map $\mathfrak{a}$ is non-empty.

Fair join pattern matching

We define the following judgements

$\mathcal{M} \models_\sigma J \text{ if } \gamma$ $\quad$ $J \text{ if } \gamma$ exactly matches $\mathcal{M}$ via substitution σ

$\mathcal{M} \models_\mathcal{I} J \text{ if } \gamma$ $\quad$ $J \text{ if } \gamma$ sparsely matches $\mathcal{M}$ via slice $\mathcal{I}$

$\mathcal{M} \models J \text{ if } \gamma \rightsquigarrow \mathcal{I}$ $\quad$ $J \text{ if } \gamma$ fairly matches $\mathcal{M}$ via slice $\mathcal{I}$

by the following inference rules:

$$\frac{\text{for all} i \in \{1, \ldots, n\} : \ \mu_i\sigma = m_i \quad \gamma\sigma}{m_1 \cdot \ldots \cdot m_n \ \models_\sigma \ \mu_1 \wedge \ldots \wedge \mu_n \text{ if } \gamma}$$

$$\frac{\mathcal{M}[\mathcal{I}] \models_\sigma J \text{ if } \gamma}{\mathcal{M} \models_\mathcal{I} J \text{ if } \gamma}$$

$$\frac{\mathcal{M} \models_\mathcal{I} J \text{ if } \gamma \quad \text{for all } \mathcal{I}' : (\mathcal{M} \models_{\mathcal{I}'} J \text{ if } \gamma \quad \text{implies} \quad \mathcal{I} \leqslant_{\text{lex}} \mathcal{I}')}{\mathcal{M} \models J \text{ if } \gamma \rightsquigarrow \mathcal{I}}$$

Matching of join definitions

Theorem

Let $J = \mu_1 \wedge \ldots \wedge \mu_p$, *then* $\mathcal{M} \models J$ if $\gamma \rightsquigarrow \mathcal{I}$ *if and only if the fair resolution* a *of* $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ *is such that* $\mathcal{I} = \mathsf{a}(1) \cdot \ldots \cdot \mathsf{a}(p)$.

Matching of join definitions

Theorem

Let $J = \mu_1 \wedge \ldots \wedge \mu_p$, *then* $\mathcal{M} \models J$ if $\gamma \rightsquigarrow \mathcal{I}$ *if and only if the fair resolution* a *of* $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ *is such that* $\mathcal{I} = \mathrm{a}(1) \cdot \ldots \cdot \mathrm{a}(p)$.

Let D $= J_1$ if $\gamma_1, \ldots, J_n$ if γ_n a sequence of join patterns and

$$\mathit{Matches} = \left\{ (\mathcal{I}, i) \;\middle|\; i \in \{1, \ldots, n\} \text{ and } \mathcal{M} \models J_i \text{ if } \gamma_i \rightsquigarrow \mathcal{I} \right\}$$

Matching of join definitions

Theorem

Let $J = \mu_1 \wedge \ldots \wedge \mu_p$*, then* $\mathcal{M} \models J$ if $\gamma \rightsquigarrow \mathcal{I}$ *if and only if the fair resolution* a *of* $\mathcal{T}(\mathcal{M}, J \text{ if } \gamma)$ *is such that* $\mathcal{I} = \mathrm{a}(1) \cdot \ldots \cdot \mathrm{a}(p)$.

Let $\mathrm{D} = J_1$ if $\gamma_1, \ldots, J_n$ if γ_n a sequence of join patterns and

$$\mathit{Matches} = \left\{ (\mathcal{I}, i) \;\middle|\; i \in \{1, \ldots, n\} \text{ and } \mathcal{M} \models J_i \text{ if } \gamma_i \rightsquigarrow \mathcal{I} \right\}$$

Another judgement induced by the rule

$$\frac{(\mathcal{I}, i) \in \mathit{Matches} \qquad \forall (\mathcal{I}', i') \in \mathit{Matches} : \; \mathcal{I} <_{\text{slex}} \mathcal{I}' \text{ or } \left(\mathcal{I} =_{\text{slex}} \mathcal{I}' \text{ and } i \leq i'\right)}{\mathcal{M} \models J_1 \text{ if } \gamma_1, \ldots, J_n \text{ if } \gamma_n \rightsquigarrow \mathcal{I}, i}$$

If $\mathcal{M} \models \mathrm{D} \rightsquigarrow \mathcal{I}, i$ then we say that D fairly matches mailbox $\mathcal{M}$ via slice $\mathcal{I}$ at i

Implementing the tree-based algorithm

The basic data structure are matching trees

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

loop: incrementally extend the assignment tree in depth-first order

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

loop: incrementally extend the assignment tree in depth-first order
for each assignment of a node X at level of the pattern's size

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

loop: incrementally extend the assignment tree in depth-first order
  for each assignment of a node X at level of the pattern's size
    if the assignment satisfies the guard
    - report the match
    - remove the matched messages from the assignment tree

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

loop: incrementally extend the assignment tree in depth-first order
  for each assignment of a node X at level of the pattern's size
    if the assignment satisfies the guard
      - report the match
      - remove the matched messages from the assignment tree
    else
      - prune X

Implementing the tree-based algorithm

The basic data structure are matching trees

used to computer assignment trees starting from the empty assignment tree

loop: incrementally extend the assignment tree in depth-first order
  for each assignment of a node X at level of the pattern's size
    if the assignment satisfies the guard
      - report the match
      - remove the matched messages from the assignment tree
    else
      - prune X
      - wait for a new message
      - goto loop.

References I

[1] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*.
MIT Press, Cambridge, MA, USA, 1986.

[2] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines.
JACM, 30(2):323–342, 1983.

[3] Alex Coto, Roberto Guanciale, and Emilio Tuosto. An abstract framework for choreographic testing.
In *ICE*, volume abs/2009.07990, 2020.

[4] Alex Coto, Roberto Guanciale, and Emilio Tuosto. Choreographic development of message-passing applications - A tutorial.
In Simon Bliudze and Laura Bocchi, editors, *Coordination Models and Languages - 22nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15-19, 2020, Proceedings*, volume 12134 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2020.

[5] Alex Coto, Roberto Guanciale, and Emilio Tuosto. An abstract framework for choreographic testing.
Journal of Logic and Algebraic Methods in Programming, 123:100712, 2021. Extended version of [3].

[6] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata.
In *ESOP 2012*, pages 194–213, 2012.

[7] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process.
J. ACM, 32(2):374–382, apr 1985.

[8] Cedric Fournet and George Gonthier. The reflexive CHAM and the join-calculus.
In *Conference Record of POPL '96: The* 23^{rd} *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 372–385, St. Petersburg Beach, Florida, January 1996.

References II

[9] Leonardo Frittelli, Facundo Maldonado, C. Hernán Melgratti, and Emilio Tuosto. A Choreography-Driven Approach to APIs: the OpenDXL Case Study.
In *COORDINATION*, volume 12134 of *LNCS*. Springer, June 2020.

[10] Roberto Guanciale and Emilio Tuosto. An abstract semantics of the global view of choreographies.
In *Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016.*, pages 67–82, 2016.

[11] Roberto Guanciale and Emilio Tuosto. Realisability of pomsets.
Journal of Logic and Algebraic Methods in Programming, 108:69–89, 2019.

[12] Roberto Guanciale and Emilio Tuosto. Pomcho: a tool chain for choreographic design.
Science of Computer Programming, 202:102535, 2021.

[13] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Fair join pattern matching for actors.
In Jonathan Aldrich and Guido Salvaneschi, editors, *38th European Conference on Object-Oriented Programming, ECOOP 2024, Vienna, Austria, September 16-20, 2024*, volume 313 of *LIPIcs*, pages 17:1–17:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[14] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Fair join pattern matching for actors (artifact).
Dagstuhl Artifacts Ser., 10(2):8:1–8:3, 2024.

[15] Carl Hewitt, Peter Boehler Bishop, and Richard Steiger. A Universal Modular ACTOR Formalism for Artificial Intelligence.
In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, USA, August 20-23, 1973*, pages 235–245. William Kaufmann, 1973.

[16] Ayman Hussein, Philipp Haller, Ioannis Karras, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Joinactors: A modular library for actors with join patterns.
Art Sci. Eng. Program., 11(1), 2026.

References III

[17] Nickolas Kavantzas, Davide Burdett, Gregory Ritzinger, Tony Fletcher, and Yves Lafon. Web services choreography description language version 1.0.
http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217. Working Draft 17 December 2004.

[18] Roland Kuhn and Alan Darmasaputra. Behaviorally typed state machines in typescript for heterogeneous swarms.
In René Just and Gordon Fraser, editors, *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, pages 1475–1478. ACM, 2023.

[19] Roland Kuhn, Hernán C. Melgratti, and Emilio Tuosto. Behavioural types for local-first software.
In Karim Ali and Guido Salvaneschi, editors, *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States*, volume 263 of *LIPIcs*, pages 15:1–15:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[20] Roland Kuhn, Hernán C. Melgratti, and Emilio Tuosto. Behavioural types for local-first software (artifact).
Dagstuhl Artifacts Ser., 9(2):14:1–14:5, 2023.

[21] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From Communicating Machines to Graphical Choreographies.
In *POPL15*, pages 221–232, 2015.

[22] Hubert Plociniczak and Susan Eisenbach. Jerlang: Erlang with joins.
In *Coordination Models and Languages: 12th International Conference, COORDINATION 2010, Amsterdam, The Netherlands, June 7-9, 2010. Proceedings 12*, pages 61–75. Springer, 2010.

[23] Carlos G. Lopez Pombo, Agustín E. Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
In Erika Ábrahám, Clemens Dubslaff, and Silvia Lizeth Tapia Tarifa, editors, *Theoretical Aspects of Computing – ICTAC 2023*, pages 119–138. Springer, 2023.

References IV

[24] Carlos López Pombo, Agustín E. Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
TCS, 1043:115247, 2025.

[25] John A. Trono. A new exercise in concurrency.
SIGCSE Bull., 26(3):8–10, September 1994.

[26] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies.
Journal of Logic and Algebraic Methods in Programming, 95:17–40, 2018. Revised and extended version of [10]. available at http://www.cs.le.ac.uk/people/et52/jlamp-with-proofs.pdf.