A Separation Logic for Resource Distribution
- extended abstract -

Nicolas Biri and Didier Galmiche

LORIA - Université Henri Poincaré
54506 Vandœuvre-lès-Nancy Cedex, France
biri{galmiche}@loria.fr

Abstract We define a separation logic (BI-Loc) that is an extension of the Bunched Implications (BI) logic with a modality for locations. Moreover, we propose a general data structure, called resource tree, that is a node-labelled tree in which nodes contain resources that belong to a partial monoid. We also define a resource tree model for this logic that allows to reason and prove properties on resource trees. We study the decidability by model checking of the satisfaction and the validity in this separation logic and also introduce a sequent calculus for deciding validity by deduction w.r.t. a resource model. Then, we relate the separation logic and resource trees to some applications and finally define a sequent calculus for BI-Loc dedicated to a theorem proving approach.

1 Introduction

The notion of *resource* is a basic one in many fields, including computer science. The location, ownership, distribution, access to and, indeed, consumption of, resources are central concerns in the design of systems (such as networks) and also of programs which access memory and manipulate data structures (such as pointers). The logic of Bunched Implications (BI), that is a resource-aware logic in which additive ($\wedge$,$\rightarrow$) and multiplicative connectives ($*, -\!*$) cohabit [14,15], appears as a kernel logic of resource (de)composition as illustrated by separation logics [12,13,16] and also by spatial logics for distributed and mobile computations [8]. It is natural, in this setting, to consider primitive notions of location or distribution and then to define a new separation logic, based on BI logic extended with a location modality, for reasoning on distributed resources. In addition, we aim to define a resource tree model for this logic that allows to reason and prove properties on based-on resource trees structures.
In this context, it is important to define models that well reflect structures but also logics that are enough expressive to represent data properties and enough restricted to decide if a given model satisfies a formula or if some properties entail other properties. We observe in recent works on tree models (and their related logics) some limits for information representation such as the difficulty to represent complex data inside nodes [4] and the choice of a resource composition that is not partial [7], knowing that partiality enables to ensure that substructures are disjoints. Therefore, we study a new data model, called resource tree, that is a labelled tree structure in which nodes contain resources that are elements of a partially defined monoid and we show how and why it is an appropriate proposal to deal with resource distribution. Then, we define a tree structure that represents complex information, through resources inside nodes, and also distinguishes the structure and the information it contains. Moreover, we can alter an existing structure and characterize inconsistent trees (not valid according to a specification).
In order to define a logic for resource trees we consider the extension of the BI logic with a modality for locations (BI-Loc). It can be viewed as a separation and spatial logic: the BI's multiplicatives naturally introduce the notion of resource separation and the location modality allows to gather

resources in some locations and thus to introduce a notion of spatial distribution of resources. The use of a partial composition on resources and the introduction of a location modality in a resource-aware logic [10] are key points in this work.
In section 2, we define what a resource tree is and illustrate its specific features. Section 3 presents the related separation logic BI-Loc and in order to analyze the decidability for resource trees against this logic, we show, in section 4, that BI is decidable by model-checking for partial resource monoids that satisfy some specific conditions. This result is extended to resource trees and its separation logic. Moreover, we derive a sequent calculus for the logic that is sound and complete w.r.t. an interpretation in terms of satisfaction. In section 5, we consider some applications of resource trees and BI-Loc and analyze the relationships with other models and logics dealing with the notions of location [4] and hierarchy of resources [1]. To complete these results, we propose, in section 6, a sequent calculus for BI-Loc that allows to check the validity of a formula for all models and then by a theorem proving approach, starting from some recent results about decidability of BI and a new complete resource semantics [11]. Section 7 includes conclusions and perspectives.

2 Resource Trees

Our goal is to define an abstract model for reasoning about distributed resources. Such a model must be rich enough to represent complex information and to manage the information distribution. We want to divide the space into locations and to explicitly mention where the resources are. In this context, it seems natural to divide a location into sub-locations and them to provide a hierarchical representation of the space and thus define the *resource tree* notion.
A resource tree is a finite tree with labelled nodes, each node containing resources which belong to a partial monoid of resources. In a resource tree, a path (list of label names) always leads to at most one unique node. More formally, a resource tree can be seen as a pair (m, f) where m is the resource contained in a root node and f is a partial function which associates a resource tree to some labels. For instance, $(m, (l \mapsto (m', nil)))$ corresponds to a tree which contains the resource m and a child at location l which contains m'. Formally, resource trees can be defined as follows:

Definition 1 (Resource Tree (1)). *Given an enumerable set of names $\mathcal{L}$ and a partial resource monoid $\mathcal{M} = (M, \bullet, e, \sqsubseteq)$, the set of resource trees over $\mathcal{M}$ (denoted $\mathcal{T}_M$) is inductively defined as follows: $\mathcal{T}_M ::= M \times [\mathcal{L} \rightharpoonup \mathcal{T}_M]$.*

Moreover, the composition of resource trees (denoted $|$) can be used to combine either trees or resources. Due to the partiality of the monoid operator $\bullet$, the composition of resources m and m' (denoted $m \bullet m'$) may be undefined. Consequently, the composition of resource trees is also partially defined. Given two resource trees $T = (m, f)$ and $T' = (m', f')$, $T|T'$ is defined iff $m \bullet m'$ is defined and for any location l such that $f(l)$ and $f'(l)$ exist then $f(l)|f'(l)$ is defined. Then $T|T' = (m'', f'')$ with $m'' = m \bullet m'$ and for any location l, $f''(l)$ is equal to $f(l)|f'(l)$ if $f(l)$ and $f'(l)$ are both undefined, to $f(l)$(resp.$f'(l)$) if $f'(l)$ (resp. $f(l)$) is undefined, and is undefined if both are undefined. The direct description of resource trees by a recursive domain equation is similar in the spirit to the representation of heap models of BI [12], as used in separation logics. In related work, a different style of model representation has been used, where a grammar of terms is given together with a syntactic structural equivalence notion [4]. We choose here such a representation that provides a convenient language for writing down model elements.

Definition 2 (Resource Tree (2)). *Given an enumerable set of names $\mathcal{L}$ and a partial resource monoid $\mathcal{M} = (M, \bullet, e, \sqsubseteq)$, a resource tree over $\mathcal{M}$ (denoted T_M) is inductively defined as follows:*

$$P ::= m \mid P|P \mid [l](P) \text{ (where } m \in M \text{ and } l \in \mathcal{L}\text{).}$$

For any resource $m, m' \in M$, $m|m' \stackrel{\text{def}}{=} m \bullet m'$.

A node labelled with l which contains a subtree P is denoted $[l](P)$. The empty tree corresponds to the neutral element of the monoid (denoted e). For instance, $[l](m|[l']m')$ represents a tree with a child l containing a resource m and a child l' that contains a resource m'.
We want that a unique location corresponds to a unique path and handle composition of trees having the same label on the root node (as $[l]P|[l]Q$) by merging nodes with the same labels. Compared to other tree models [4], a major improvement is that nodes are not only labels but can also contain information (resources). Moreover, we have a unique corresponding location for a given path and the composition operator is partial. There is a structural equivalence $\equiv$ between resource trees that is defined as follows:

- $P \equiv P$	- if $P \equiv Q$ then $Q \equiv P$	- if $P \equiv Q$ and $Q \equiv R$ then $P \equiv R$
- $P\|Q \equiv Q\|P$	- $(P\|Q)\|R \equiv P\|(Q\|R)$	- if $P \equiv Q$ then $P\|R \equiv Q\|R$
- $P\|e \equiv P$	- $[l]P\|[l]Q \equiv [l](P\|Q)$	- if $P \equiv Q$ then $[l]P \equiv [l]Q$

The rule $([l]P|[l]Q \equiv [l](P|Q))$ corresponds to the way we handle the composition of trees in case we must compose two sibling nodes with the same label. In a nutshell, the composition operator ($|$) merges nodes with the same label and composes others as the usual composition of trees. Then, the composition of two nodes with the same labels is equivalent to one node which contains resources and subtrees of these two nodes, as illustrated in the figure 1.

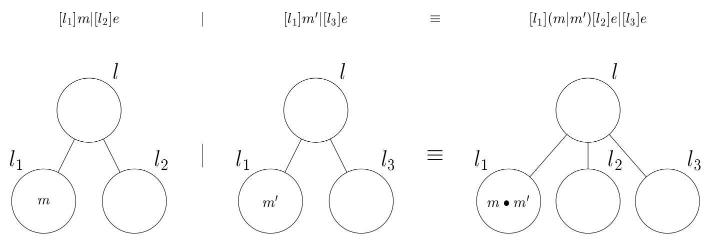
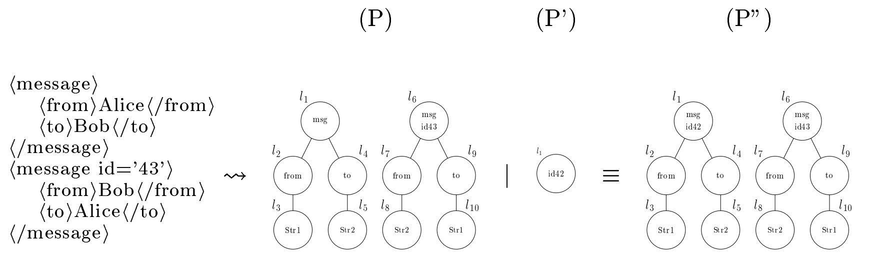


Figure 1. Resource Tree Composition

Moreover, we can inductively extend the $\sqsubseteq$ relation of the monoid to resource trees by defining the partial ordering relation $\preccurlyeq$ on resource trees is follows: *1)* $m \preccurlyeq Q$ *if and only if* $Q \equiv m'$ *and* $m \sqsubseteq m'$*; 2)* $[l]P' \preccurlyeq Q$ *if and only if* $Q \equiv [l]Q'$ *and* $P' \preccurlyeq Q'$ *and 3)* $P'|P'' \preccurlyeq Q$ *if and only if* $Q \equiv Q'|Q''$,$P' \preccurlyeq Q'$ *and* $P'' \preccurlyeq Q''$.

Let us note that a resource tree P defined with Definition 2 corresponds to the following resource tree $\tilde{P}$ with the Definition 1: $m \rightsquigarrow (m, nil)$, $P|P' \rightsquigarrow \tilde{P}|\tilde{P'}$ and $[l]P \rightsquigarrow (e, l \mapsto \tilde{P})$. Then, we can prove that, *given two resource trees* P *and* P' *defined with Definition 2, we have* $P \equiv P'$ *if and only if* $\tilde{P} = \tilde{P'}$.

We now illustrate some specific features of resource trees by considering the representation of an XML tree as a resource tree. Actually, we may represent the tree structure and the node contents as resources. The resource tree composition allows to alter a tree structure but in a fine way. Actually one can compose, with the $|$ operator, an existing tree with another one which will add a resource subtree into a specific node. However, this property implies a less intuitive representation of information. In fact, information which occurs through labels in [4] does not correspond to labels in resource trees but is represented by resources inside nodes. It provides an easier way to represent complex information and distinguishes the shape of data (the tree structure) from information it contains (tags and attributes). In our data, a location name only refers to a specific part of the tree structure.



$P \equiv [l_1](msg|[l_2](from|[l_3]Str1)|[l_4](to|[l_5]Str2))|[l_6](msg|id43|[l_7](from|[l_8]Str1)|[l_9](to|[l_{10}]Str2))$
$P' \equiv [l_1]id42$
$P'' \equiv [l_1](msg|id42|[l_2](from|[l_3]Str1)|[l_4](to|[l_5]Str2))|[l_6](msg|id43|[l_7](from|[l_8]Str1)|[l_9](to|[l_{10}]Str2))$

Figure 2. Representing and Altering a XML Data

Figure 2 gives a representation of a XML tree and illustrates how to add an attribute to a given node. We have the representation of the XML file on the left-hand side and the second tree shows how we can add the attribute *id = '42'* to the node corresponding to the first message. Let us note that attributes and tags are both represented as resources. Our representation of attributes allows us to describe pointer references (references to another part of the tree using the *id* attribute) such as exposed in [6]. Moreover, we can as well add a subtree at a given node instead of just one resource. Let us suppose that, instead of adding the attribute to the first message, we add it to the second one and that also, as it is the case in XML, the *id* attribute is unique (i.e., you cannot declare two *id* attributes in the same node). Consequently, the resulting tree does not correspond to a valid XML data. The partiality of our resource composition gives an easy way to handle with this kind of invalid data. Actually, it is sufficient to declare that any composition of a resource representing an *id* data with another one representing another *id* data is undefined.
Compared to edge-labelled trees [4], resource trees allow representation of information inside nodes where the former can only represent tag names and tree structure. With the composition operator, we can alter an existing structure instead of just to add information next to the existing one, and characterize inconsistent trees (trees which are not valid according to a specification).

3 A New Separation Logic

The BI logic is a resource-aware separation logic in which additives ($\wedge$,$\rightarrow$) and multiplicatives ($*, -\!\!*$) cohabit. The decidability of propositional BI logic has been recently proved and a based-on partial monoid semantics, that is complete, has been defined for intuitionistic additives [11]. We define here an extension of BI with a modality of location (BI-Loc) and a based-on resource tree model for this logic, according to the satisfaction relation $P \models \phi$. Then, we define what a valid formula is and we show the correspondence between validity and satisfaction relation.

Definition 3. *Given a set of location names $\mathcal{L}$ and a language of propositional letters $\mathcal{P}$, the set of formulae of BI-Loc is inductively defined as follows:*

$$\phi ::= [l]\phi \mid p \mid I \mid \phi * \phi \mid \phi -\!\!* \phi \mid \top \mid \bot \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \rightarrow \phi \ (\textit{with } l \in \mathcal{L} \textit{ and } p \in P)$$

The modality [-] is a spatial modality as discussed above. We now define a resource tree model for this logic based on the *partial monoid* resource model [11].

Definition 4 (Interpretation). *Given a partial monoid of resources $\mathcal{M} = \langle M, \bullet, e, \sqsubseteq\rangle$ and a language $\mathcal{P}$ of propositional letters, an interpretation is a function $[\![\,]\!]: \mathcal{P} \to \wp(M)$.*

Definition 5 (Resource Tree Model). *A resource tree model is a triplet $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models\rangle$ such that $\mathcal{M} = (M, \bullet, e, \sqsubseteq)$ is a partial resource monoid, $[\![\,]\!]$ is an interpretation and $\models$ is a forcing relation over $T_M \times \mathcal{P}(P)$ (T_M is the set of resource trees over M and $\mathcal{P}(P)$ is the set of BI-Loc propositions) that satisfies*

- $P \models [l]\phi$ *iff there exists Q such that $[l]Q \preccurlyeq P$ and $Q \models \phi$*
- $P \models I$ *iff $e \preccurlyeq P$*
- $P \models \phi * \psi$ *iff there exist Q, R such that $Q|R$ is defined, $Q|R \preccurlyeq P$, $Q \models \phi$ and $R \models \psi$.*
- $P \models \phi \mathrel{-\!\!*} \psi$ *iff for all Q such that $Q \models \phi$ and $P|Q$ is defined, $P|Q \models \psi$.*
- $P \models \top$ *always.*
- $P \models \bot$ *never.*
- $P \models p$ *iff there exists $m \in M$ such that $m \preccurlyeq P$ and $m \in [\![p]\!]$.*
- $P \models \phi \vee \psi$ *iff $P \models \phi$ and $P \models \psi$.*
- $P \models \phi \wedge \psi$ *iff $P \models \phi$ or $P \models \psi$.*
- $P \models \phi \to \psi$ *iff $P \models \phi$ implies that $P \models \psi$.*

This semantics relies on the partial monoid semantics of BI [11], but it deals with resource trees instead of resources. Consequently, we have extended the definition of the forcing relation in order to handle such trees and the [-] modality.

Definition 6 (Satisfaction). *Given a resource tree model $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models\rangle$, a resource tree P over M satisfies a formula ϕ if and only if $P \models \phi$ holds.*

Let us clarify the meaning of some formulae, especially those involving units and the $\to$ operator. First of all, it appears that $[l]\top$ (resp. $[l]I$) is not equivalent to $\top$ (resp. I). The first formula indicates that there exists a child node called $[l]$ (which must be empty in the case of $[l]I$) whereas the second formula does not ensure such an existence. Secondly, $\bot$ does not behave as other units. Actually, $[l]\bot$ is equivalent to $\bot$ and both cannot be satisfied by a resource tree. Finally, $[l](\phi \to \phi)$ is not equivalent to $[l]\phi \to [l]\phi$. Actually, the first one is satisfied by any resource tree which have a child-node l while the second one is always satisfied. An essential point is that the structural equivalence defined in Section 2 does not alter the satisfaction relation.

Lemma 1. *Let P and Q be resource trees such that $P \equiv Q$ and ϕ be a BI-Loc formula, $P \models \phi$ if and only if $Q \models \phi$.*

Proof. By structural induction.

We can now define the validity of a BI-Loc formula:

Definition 7 (Validity on a Resource Tree Model). *Given a resource tree model $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models\rangle$ and a formula ϕ, $\mathcal{T} \models \phi$ if and only if for any resource tree P, such that its resources belong to $\mathcal{M}$, we have $P \models \phi$.*

The validity on a resource model could be expressed as a satisfaction relation.

Lemma 2. *Given a formula ϕ, $\models \phi$ if and only if $e \models \top \mathrel{-\!\!*} \phi$.*

Proof. By definition of the satisfaction relation.

Moreover, by extension, the validity of a formula is defined as follows:

Definition 8 (Validity). *Given a formula ϕ, $\models \phi$ if and only if for any resource tree model $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models \rangle$, $\mathcal{T} \models \phi$.*

Compared to the spatial logic for edge-labelled trees [4] which handles only units, our logic handles propositional letters and a location name always refers to the same location (which is neither an advantage nor a disadvantage, but just another view of locations). Our logic does not include a placement @ operator but we can use propositions and resources in order to embed the behavior of SLT locations, by adding quantifications on locations in BI-Loc, as discussed later.

4 Decidability on a Resource Model

Concerning the above-mentioned calculi (ambients, trees or pointers) and their spatial or separation logics, recent works have investigated the border between decidable and undecidable cases of model checking for a calculus and its related logic [4,5,9]. Let us recall that the model-checking problem is to decide whether a given object satisfies (is a model of) a given formula. In these logics, we observe that the decidability depends on interactions between the separation connectives ($*$, $-\!\!*$), the classical ones and the ones introducing spatial modalities. One central point is that the $-\!\!*$ operator introduces an infinite quantification on trees. In order to obtain the decidability by model-checking, we must be able to bound this quantification and to capture the interactions of $-\!\!*$ with other connectives. These key points have been already identified, but in a different way, in the proof of the decidability of propositional BI logic [11].

4.1 Deciding Validity by Model Checking in BI

A main problem is the infinite quantification introduced by the $-\!\!*$ operator. It is not directly related to the tree structure and consequently we start by focusing on the classical resource monoid and BI formulae. Therefore, we propose a restriction on the resource monoid which allows to model check the satisfaction and validity of a BI formula. We define sufficient conditions on monoids in order to decide the satisfaction by model checking.

Definition 9 (Boundable Resource Model). *A resource model is said boundable if:*

1. *For any $m \in M$ there exist $m_1, \dots m_n$ such that $m \sqsubseteq m_1 \bullet \dots \bullet m_n$ and for any m_i, m_i is not decomposable (i.e., there is no m', m'' such that $m' \neq e$, $m'' \neq e$ and $m_i \sqsubseteq m' \bullet m''$).*
2. *For every subset of propositions L' and every integer n there exists an equivalence relation $\cong_{L',n}$ such that :*
   (a) *$e \cong_{L',n} m$ iff $e \sqsubseteq m$*
   (b) *For any $m, m' \in M$ and $p \in L'$ if $m \cong_{L',n} m'$ then $m \in [\![p]\!]$ iff $m' \in [\![p]\!]$.*
   (c) *For any $m, m', m'' \in M$, if $m \cong_{L',n} m'$ and if $m \bullet m''$ is defined then $m' \bullet m''$ is defined and $m \bullet m'' \cong_{L',n} m' \bullet m''$.*
   (d) *For any $k \leq n$, $m, m' \in M$ such that $m \cong_{L',n} m'$ if there exist $m_1, \dots m_k$ such that $m \sqsubseteq m_1 \bullet \dots \bullet m_k$ then there exist $m'_1, \dots m'_k$ such that $m' \sqsubseteq m'_1 \bullet \dots \bullet m'_k$ and for any $i \leq k$ $m_i \cong_{L',0} m'_i$.*
   (e) *$M_{/\cong_{L',n}}$ is finite.*

The above definition does not impose to have a finite monoid but only requires that the monoid has a finite quotient for each equivalence relation. Some instantiations of such models satisfying these requirements are given later.
We show that, for a given BI formula, there exists a finite equivalence class to work on. Given a BI formula ϕ, we consider L_ϕ the set of propositions which occur in ϕ and we define the size of ϕ (denoted $s(\phi)$) as follows:

Definition 10. *Given a BI formula ϕ, the size of ϕ is inductively defined by*

- $s(I) = 1$	- $s(\top) = 0$	- $s(\phi' \vee \phi'') = max(s(\phi'), s(\phi''))$
- $s(p) = 1$	- $s(\phi' \mathrel{-\!\!*} \phi'') = s(\phi'')$	- $s(\phi' \wedge \phi'') = max(s(\phi'), s(\phi''))$
- $s(\bot) = 0$	- $s(\phi' * \phi'') = s(\phi') + s(\phi'')$	- $s(\phi' \rightarrow \phi'') = max(s(\phi'), s(\phi''))$

The size of a formula ϕ determines the number of resource decompositions that are necessary to decide if ϕ is satisfied by a resource or not. Our goal is to show that checking if a resource m satisfies $\phi \mathrel{-\!\!*} \psi$ corresponds to checking if $m \bullet m'$ satisfies $\phi \mathrel{-\!\!*} \psi$ for a given subset M' of M.

Lemma 3 (Monotonicity). *If $m \cong_{L',n} m'$, $L'' \subseteq L'$ and $n' \leq n$ then $m \cong_{L'',n'} m'$.*

Proof. From the conditions 2b and 2d of Definition 9.

Lemma 4. *If $m \bullet m' \cong_{L',n_1+n_2} t$ then there exist t', t'', such that $t \sqsubseteq t' \bullet t''$, $m \cong_{L',n_1} t'$ and $m' \cong_{L',n_2} t''$.*

Proof. From the conditions 2c and 2d of Definition 9.

Lemma 5. *If $m \cong_{L_\phi,s(\phi)} m'$ and $m \models \phi$ then $m' \models \phi$.*

Proof. By induction on the structure of ϕ. We develop some interesting cases.
- *Case* $\phi \equiv I$. As $m \models \phi$, $e \sqsubseteq m$ and $m' \cong_{\emptyset,1} m$, from condition 2a of a boundable monoid we deduce $e \sqsubseteq m'$, and $m' \models I$.
- *Case* $\phi \equiv \psi * \psi'$. By definition there exist m_1 and m_2 such that $m_1 \models \psi$, $m_2 \models \psi'$ and $m_1 \bullet m_2 \sqsubseteq m$. Thus by lemma 4, there exist m_1', m_2' such that $m_1 \cong_{L_\phi,s(\psi)} m_1'$ and $m_2 \cong_{L_\phi,s(\psi')} m_2'$. By induction hypothesis, we can conclude.
- *Case* $\phi \equiv \psi \mathrel{-\!\!*} \psi'$. We consider m_1 such that $m_1 \models \psi$ then $m \bullet m_1 \models \psi'$. As $m \cong_{L_\phi,s(\phi)} m'$, we have $m \bullet m_1 \cong_{L_\phi,s(\phi)} m' \bullet m_1$ (due to condition 2c of a boundable monoid). We conclude with the induction hypothesis.

Corollary 1. *For any resource m and any formula of the form $\phi \mathrel{-\!\!*} \psi$, $m \models \phi \mathrel{-\!\!*} \psi$ iff for any $m' \in M_{/\cong_{L_\phi,s(\phi)}}$ such that $m \bullet m'$ is defined, $m \bullet m' \models \psi$.*

Proof. Immediate consequence of Lemma 5 and condition 2 of Definition 9.

As the case of the $\mathrel{-\!\!*}$ operator is the crux of the satisfaction problem, we have

Theorem 1 (Satisfaction Decidability for BI). *Given a boundable partial resource monoid $\mathcal{M} = (M, \bullet, e, \sqsubseteq)$, for any BI formula ϕ and any resource $m \in M$, $m \models \phi$ is decidable.*

Proof. Consequence of Corollary 1 (for the finite test to check formula of the form $\phi \mathrel{-\!\!*} \psi$) and of condition 1 of a boundable resource model (to ensure that there is only a finite set of decompositions necessary to check a formula as $\phi * \psi$).

Corollary 2 (Validity Decidability for BI). *Given a boundable partial resource monoid $\mathcal{M} = (M, \bullet, e, \sqsubseteq)$, for any BI formula ϕ, $\models_{\mathcal{M}} \phi$ is decidable.*

Proof. Immediate consequence of Theorem 1 using Lemma 2.

4.2 Validity by Model Checking in BI-Loc

In order to extend the above results to BI-Loc and resource trees we must add the tree structure without loosing the ability to bound the resource tree when we consider the $-\!\!*$ operator. Therefore, we have two main problems: 1) to restrict the location names to deal with since an infinity of locations leads to an infinity of trees and 2) to bound the height of the trees.
Given a BI-Loc formula ϕ, $\mathcal{L}_\phi$ denotes the finite set of location names of ϕ.

Lemma 6. *Given a formula ϕ, a resource tree P and two location names l and l' which do not belong to $\mathcal{L}_\phi$, $P \models \phi$ if and only if $P\{^l/_{l'}\} \models \phi$.*

Proof. By structural induction on ϕ.

Corollary 3. *For any BI-Loc formula ϕ and any resource tree P, there exists a resource tree P' such that the location names that occur in P' belong to $\mathcal{L}_\phi \cup \{l_\pi\}$ (where l_π is a location name which does not occur in ϕ) and we have $P \models \phi$ if and only if $P' \models \phi$.*

Proof. We replace all occurrences of locations in P which are not in ϕ by l_π and obtain P'.

For a given resource tree P, we note $h(P)$ its height which is defined as usual for a tree. Moreover, we define the height of a BI-Loc formula.

Definition 11 (Height of a Formula). *Given a BI-Loc formula ϕ, the height $h(\phi)$ of ϕ is inductively defined as:*

- $h(I) = 1$	- $h([l]\phi') = 1 + h(\phi')$	- $h(\phi' \vee \phi'') = max(h(\phi'), h(\phi''))$
- $h(p) = 1$	- $h(\phi' -\!\!* \phi'') = h(\phi'')$	- $h(\phi' \wedge \phi'') = max(h(\phi'), h(\phi''))$
- $h(\bot) = 0$	- $h(\top) = 0$	- $h(\phi' \rightarrow \phi'') = max(h(\phi'), h(\phi''))$
		- $h(\phi' * \phi'') = max(h(\phi'), h(\phi''))$

The height of a formula is the maximum height of the trees we have to consider in order to check if a formula is valid or not. Therefore, we have the following result:

Proposition 1. *For any BI-Loc formula ϕ and any resource tree P, there exists a resource tree P' such that $h(P') \leq h(\phi)$ and $P \models \phi$ if and only if $P' \models \phi$.*

Proof. The interesting case occurs when $h(P) > h(\phi)$. It is sufficient to *cut* subtrees which are *too high* and to prove, by structural induction on ϕ, that the resulting tree satisfies the condition.

Contrary to what is done in [4], we do not require to limit the width of the tree. This is due to the rule $([l]m)|([l]m') \equiv [l](m|m')$. Consequently, the width of the tree is limited by the number of locations name which are relevant.

Theorem 2 (Satisfaction Decidability). *Given a resource tree model $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models \rangle$ where $\mathcal{M}$ is a boundable partial resource monoid, for any BI-Loc formula ϕ and any resource tree $P \in \mathcal{T}$, $P \models \phi$ is decidable.*

Proof. We can extend the congruence relations of a boundable monoid in order to include restrictions on height and location names introduced by the last lemmas. Then, we deal with the set of trees modulo this relation and prove that this set is finite.

Corollary 4 (Validity Decidability). *Given a resource tree model $\mathcal{T} = \langle \mathcal{M}, [\![\,]\!], \models \rangle$ where $\mathcal{M}$ is a boundable partial resource monoid, for any BI-Loc formula ϕ, $\mathcal{T} \models \phi$ is decidable.*

Proof. Immediate consequence of Theorem 2 with Lemma 2.

In order to build a sequent calculus to deduce the validity of a formula from others, we can define a sequent calculus with contexts as in [4]. A context Γ or Δ is a finite multiset of expressions $P : \phi$ where P is a tree and ϕ is a formula. We obtain such a calculus by only adding the following rules to the sequent calculus of [4].

$$\frac{P \cong_{L_\phi, s(\phi)} Q}{\Gamma, P : \phi \vdash Q : \phi, \Delta}\ (\mathrm{Id}) \qquad \frac{\forall Q, R. P \equiv Q|R \Rightarrow \Gamma, Q : \phi, R : \psi \vdash \Delta}{\Gamma, P : \phi * \psi \vdash \Delta}\ (* : \mathrm{L})$$

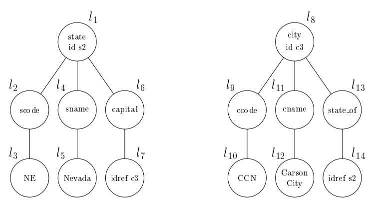
$$\frac{\forall m. P \equiv m \Rightarrow m \notin [\![p]\!]}{\Gamma, P : p \vdash \Delta}\ (\mathrm{Prop : L}) \qquad \frac{\exists m. m \equiv P \ and \ m \notin [\![p]\!]}{\Gamma \vdash P : p, \Delta}\ (\mathrm{Prop : R})$$

The interpretation $[\![P_1 : \phi_1, \ldots, P_n : \phi_n \vdash Q_1 : \psi_1, \ldots Q_m : \psi_m]\!]$ of a sequent is defined as follows: if $P_1 \models \phi_1, \ldots, P_n \models \phi_n$ then $Q_1 \models \psi_1, \ldots, Q_m \models \psi_m$. We can prove by structural induction that this calculus is sound and complete according to the semantics of resource trees and also that *the existence of a derivation of a sequent $\Gamma \vdash \Delta$ is decidable.*

5 Resource Trees and Applications

XML trees. The example of section 2 presents a resource tree representation of an XML entry, in which location names are arbitrary and do not have a particular meaning. In order to make abstraction of location names, it looks useful to provide quantifications on locations in BI-Loc. In this context, we need standard quantifications to state that there exists a location where a proposition is true and that a proposition is true for all locations. Moreover, we extend the forcing relation of the resource trees as follows: $P \models \exists l.\phi$ if and only if there exists l' such that $P \models \phi\{^{l'}/_{l}\}$ and $P \models \forall l.\phi$ if and only if for all l', $P \models \phi\{^{l'}/_{l}\}$. Such quantifications, mixed with the expressivity of resource trees, allow to treat useful data structures knowing that they may lead to undecidability if their use is not restricted enough.
Let us consider how we can represent "pointers" in trees, such as proposed in [6]. Actually, if we consider the XML example and a *corresponding* resource tree proposed below, we observe that the content of a *capital* tag must refer to a *city* and the content of the *state_of* tag must refer to a *state*.

⟨state id='s2'⟩
    ⟨scode⟩NE⟨/scode⟩
    ⟨sname⟩Nevada⟨/sname⟩
    ⟨capital⟩idref='c3'⟨/capital⟩
⟨/state⟩
⟨city id='c3'⟩
    ⟨ccode⟩CCN⟨/ccode⟩
    ⟨cname⟩Carson City⟨/cname⟩
    ⟨state_of⟩idref='s2'⟨/state_of⟩
⟨/city⟩



In our model, to represent and check the property of the link between *capital* and a *city* for a given *id* corresponds to check the satisfaction of the following formula:

$$\forall l_1.(\exists l_2.(\exists l_3.([l_1](state * [l_2](capital * [l3](content \wedge id))) * \top)) \rightarrow \exists l_4.([l_4](city * id) * \top)$$

Here *state* (resp. *capital*) means that the location contains a *state* (resp. *capital*) tag. *Content* and *id* respectively indicate that the node describes the content of above tag and that a resource corresponding to the given *id* is present. Thus, *content* $\wedge$ *id* means that the content of the tag is the *id* we are looking for. With a similar approach it is possible to check that two nodes do not have the same *id*.

Hierarchical Storage. Ahmed et al. have recently proposed a logic for reasoning about logical storage [1], that is inspired by the BI logic and the Ambients logic. As our work is also inspired by the former, it seems natural to analyze if our proposal can be used and adapted to reason about the hierarchical storage. Let us consider the store semantics of [1]. Let Val be an abstract set of values, a store is a partial map from paths p (a list of locations) to values v: $Store \quad s \in Path \rightharpoonup Val$. An intuitive relation arise between $Path$ and our locations and between Val and our monoid of resources. Then, the store semantics can be seen as a specific resource tree model, the path name of this semantics corresponding to the set of locations. Similarly, Val corresponds to a monoid such that the neutral element is the null value. The composition relation is totally undefined except for composition with the neutral element (we cannot have two values in a given store). However, BI-Loc is propositional (not first-order) and does not provide operators to check adjacency of stores. Using the resource tree and BI-Loc to reason about hierarchical storage is consequently less expressive but it has other advantages. Actually, if we restrict the propositions to type verification (the store contains a value of a given type), the resource model satisfies the requirements for being model checked and therefore we can check validity of BI-Loc formulae in the store semantics.

Spatial logics for Trees. Having previously compared resource trees with other tree models, we now propose an extension of our logic to encode the spatial logic for trees. In [4] one can have two sibling edges with the same label. So, $n[0]|n[0]$ is a tree with two sibling nodes accessible through a path n. A corresponding resource tree can be $[l_1]\overline{n} * [l_2]\overline{n}$ where $\overline{n}$ is the result of a mapping from the set of location names of the edge labelled-tree into the monoid of a given set of resources. But such a translation provides new arbitrary location names in the resulting resource tree. Thus if we aim to find a BI-Loc formula ϕ such that the resulting tree satisfies ϕ, we have to handle with those locations. But the standard quantifications do not behave at it should to be. Actually, a translation of the formula $l[0]|l[0]$ should intuitively be $\exists l_1.\exists l_2.[l_1]I * [l_2]I$. However, with standard quantifications, we cannot ensure that l_1 and l_2 are distinct. Then, we must restrict the quantifications to ensure that existential quantification do not make equal two locations which must be distinct. This point is related to the general problem of label (or name) restriction in our model we will develop in further work. With a fresh label quantification, the corresponding BI-Loc formula of $n[\phi]$ (called location) would be $\exists l.[l](\overline{n} * \overline{\phi})$ (where $\overline{\phi}$ is the corresponding formula of ϕ). As in the previous part about XML, we hope that the restricted use of quantifications will not alter some decidability results. Moreover, we can show that a formula of SLT logic, that includes the placement formula denoted $\phi@n$, can be transformed into a SLT formula without @ and this translation does not alter the satisfaction relation defined for edge-labelled trees.

6 Decidability for All Resource Models

Reasoning about validity of a formula for all resource tree models cannot be done by model checking but by theorem proving. The sequent calculus with contexts of the previous section is not a solution because contexts require to work on a given model. Propositional BI has been proved decidable by theorem proving [11] and there exists several calculi [10,11,15] to check validity of BI formulae. Like for model checking, we can extend the decidability result to BI-Loc with some sequent calculus rules that handle locations and also consider extensions of the related proof-search methods.
A sequent $\Gamma \vdash \phi$ is valid if and only if for any resource tree model T and for all $P \in T$, $P \models \phi_\Gamma$ implies $P \models \phi$. The sequent calculus for BI-Loc is given in Figure 3. The rules are those of BI plus the (Loc) rule that handles the location modality. In this rule, $[l]\Gamma$ means that each formula of the bunch has the form $[l]\phi$. It can be read bottom up as *resource trees which satisfy $[l]\Gamma$ also satisfy $[l]\phi$ if and only if resource trees satisfying Γ also satisfy ϕ.* The non-emptiness condition

$$\frac{}{\phi \vdash \phi}\ (\mathrm{ax}) \quad \frac{\Delta \vdash \phi}{\Gamma \vdash \phi}\ (\Delta \equiv \Gamma) \quad \frac{\Gamma(\Delta) \vdash \phi}{\Gamma(\Delta; \Delta') \vdash \phi}\ (\mathrm{w}) \quad \frac{\Gamma(\Delta; \Delta) \vdash \phi}{\Gamma(\Delta) \vdash \phi}\ (\mathrm{w})$$

$$\frac{}{\Gamma(\bot) \vdash \phi}\ (\bot_L) \quad \frac{\Gamma(\emptyset_m) \vdash \phi}{\Gamma(I) \vdash \phi}\ (I_L) \quad \frac{}{\emptyset_m \vdash I}\ (I_R) \quad \frac{\Gamma(\emptyset_a) \vdash \phi}{\Gamma(\top) \vdash \phi}\ (\top_L) \quad \frac{}{\emptyset_a \vdash \top}\ (\top_R)$$

$$\frac{\Gamma(\phi, \phi') \vdash \psi}{\Gamma(\phi * \phi') \vdash \psi}\ (*_L) \quad \frac{\Gamma \vdash \phi \quad \Delta \vdash \phi'}{\Gamma, \Delta \vdash \phi * \phi'}\ (*_R) \quad \frac{\Gamma \vdash \phi \quad \Delta(\Delta', \phi') \vdash \psi}{\Delta(\Delta', \Gamma, \phi -\!\!* \phi') \vdash \psi}\ (-\!\!*_L) \quad \frac{\Gamma, \phi \vdash \phi'}{\Gamma \vdash \phi -\!\!* \phi'}\ (-\!\!*_R)$$

$$\frac{\Gamma(\phi; \phi') \vdash \psi}{\Gamma(\phi \wedge \phi') \vdash \psi}\ (\wedge_L) \quad \frac{\Gamma \vdash \phi \quad \Delta \vdash \phi'}{\Gamma; \Delta \vdash \phi \wedge \phi'}\ (\wedge_R) \quad \frac{\Gamma \vdash \phi \quad \Delta(\Delta'; \phi') \vdash \psi}{\Delta(\Delta'; \Gamma; \phi \to \phi') \vdash \psi}\ (\to_L)$$

$$\frac{\Gamma(\phi) \vdash \psi \quad \Delta(\phi') \vdash \psi}{\Gamma(\phi \vee \phi'); \Delta(\phi \vee \phi') \vdash \psi}\ (\vee_L) \quad \frac{\Gamma \vdash \phi_i (i = 1, 2)}{\Gamma \vdash \phi_1 \vee \phi_2}\ (\vee_R) \quad \frac{\Gamma; \phi \vdash \phi'}{\Gamma \vdash \phi \to \phi'}\ (\to_R)$$

$$\frac{\Gamma \vdash \phi}{[l]\Gamma \vdash [l]\phi}\ (\mathrm{Loc})\text{with } [l]\Gamma \text{ not an empty bunch}$$

Figure 3. A Sequent Calculus for BI-Loc

in the (*Loc*) rule is required to ensure that the location l exists in the resource tree. Without this requirement, we can prove the sequent $\vdash [l](\phi \to \phi)$ which is false since it is not valid for a tree which does not contain a location l (as $P \models [l]\psi$ if and only if there exists P' such that $P \preccurlyeq [l]P'$ and $P' \models \psi$).

Theorem 3 (Soundness and Completeness). *There exists a proof of $\Gamma \vdash \phi$ in the BI-Loc sequent calculus if and only if $\models \Gamma -\!\!* \phi$ is valid.*

Proof. As BI is sound and complete for the partial monoid semantics [11], we show that sequent rules inherited from BI are sound according to the resource tree semantics. Then, we prove that the (*Loc*) rule is also sound. Given a sequent $\Gamma \vdash \phi$ and a resource tree P such that $P \models \phi_\Gamma$ (where ϕ_Γ represents any formula in Γ) and $P \models \phi$ then, by definition $[l]P \vdash [l]\phi_\Gamma$ and $[l]P \vdash [l]\phi$. The completeness proof can be done by proving that if $\Gamma \not\vdash \phi$ then there exists P such that $P \models \phi_\Gamma$ and $P \not\models \phi$.

Proof search for BI can be extended to handle locations that do not introduce infinite loops. Actually, as in the decidability by model checking, the crux of the finite model property is to bound the infinite tableau branch introduced by the $-\!\!*$ operator (see [11] for details). As the location modality does not introduce any infinite quantification on trees, we can build a finite tableau for BI-Loc. Moreover, we must study if we have criteria to decide if a given branch of a tableau is closed or not. Actually, we must ensure that a location exists for some subformula. Furthermore, as the $\top$ unit is local, we can obtain a partially closed branch (branch which is not closed for all locations). Both can be systematically handled and thus we can define more complex criterias, but we can decide if a tableau is closed or not.

7 Conclusions and Perspectives

This work emphasizes that BI logic and its based-on resource semantics [11] is a logical kernel from which spatial and separation logics are defined for reasoning about various data structures or on resource distribution and interaction. We have introduced some spatial information by fixing resources in some locations and proposed a separation and spatial logic BI-Loc from which we can describe various resource models. In this context, we introduce a particular data model, called resource tree, in which nodes contain resources that are elements of a partially defined monoid. At the present state, the representation is static because when the resources are distributed in the tree, they cannot move from their initial location. A first solution could be to extend the BI-Loc logic with movement modalities as it has been done with linear logic in the context of modeling distributed and mobile systems [2]. Our results about validity by model checking and by deduction for BI and BI-Loc appear important for a further study of validity and reasoning in new separation and spatial logics like the one dedicated to hierarchical storage [1]. In this context, it should be relevant to compare our approach with the spatial logic for concurrency [3] which also include spatial and separation aspects related to a concurrent calculus model. We aim to deeply study how to use such based-on resource models (not only trees) and related logics for program verification. Moreover, we will study how our results about theorem proving in BI logic [11] can be extended to handle distribution and mobility aspects.

References

1. A. Ahmed, L. Jia, and D. Walker. Reasoning about hierarchical storage. In *18th IEEE Symposium on Logic in Computer Science*, pages 33–43, Ottawa, Canada, 2003.
2. N. Biri and D. Galmiche. A modal linear logic for distribution and mobility. In *FLOC'02 Workshop on Linear Logic*, July 2002. Copenhagen, Denmark.
3. L. Caires and L. Cardelli. A spatial logic for concurrency (part I). In *4th Int. Symposium on Theoretical Aspects of Computer Software, TACS 2001, LNCS 2215*, pages 1–37, Sendai, Japan, October 2001.
4. C. Calcagno, L. Cardelli, and A. Gordon. Deciding validity in a spatial logic for trees. In *ACM Sigplan Workshop on Types in Language Design and Implementation, TLDI'03*, New Orleans, USA, 2003.
5. C. Calcagno, H. Yang, and P. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *21st Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'01, LNCS 2245*, pages 108–119, Bangalore, India, 2001.
6. L. Cardelli, P. Gardner, and G. Ghelli. Querying trees with pointers, 2003. Draft.
7. L. Cardelli and G. Ghelli. A query language based on the Ambient Logic. In *9th European Symposium on Programming, ESOP'01, LNCS 2028*, pages 1–22, Genova, Italy, 2001.
8. L. Cardelli and A.D. Gordon. Anytime, anywhere - modal logics for mobile ambients. In *27th ACM Symposium on Principles of Programming Languages, POPL 2000*, pages 1–13, Boston, USA, 2000.
9. W. Charatonik and J.M. Talbot. The decidability of model checking mobile ambients. In *15th Int. Workshop on Computer Science Logic, CSL 2001, LNCS 2142*, pages 339–354, Paris, France, 2001.
10. D. Galmiche and D. Méry. Connection-based proof search in propositional BI logic. In *18th Int. Conference on Automated Deduction, CADE-18, LNAI 2392*, pages 111–128, 2002. Copenhagen, Denmark.
11. D. Galmiche, D. Méry, and D. Pym. Resource Tableaux (extended abstract). In *16th Int. Workshop on Computer Science Logic, CSL 2002, LNCS 2471*, pages 183–199, Edinburgh, 2002.
12. S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *28th ACM Symposium on Principles of Programming Languages, POPL 2001*, pages 14–26, London, UK, 2001.
13. P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *15th Int. Workshop on Computer Science Logic, CSL 2001, LNCS 2142*, pages 1–19, Paris, France, 2001.
14. P.W. O'Hearn and D. Pym. The Logic of Bunched Implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.
15. D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002.
16. J. Reynolds. Separation logic: A logic for shared mutable data structures. In *17th IEEE Symposium on Logic in Computer Science*, pages 55–74, Copenhagen, Denmark, July 2002.