Status of the New Backend

Karim Abdelhak, Bernhard Bachmann
University of Applied Sciences Bielefeld
Bielefeld, Germany

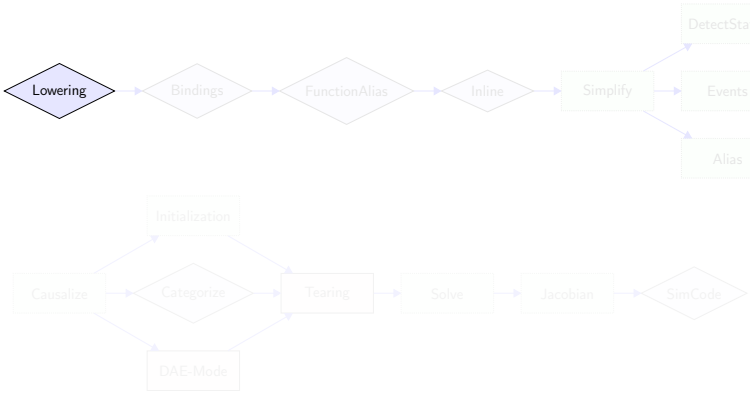
February 2, 2026

Section 1

Overview

Backend Modules

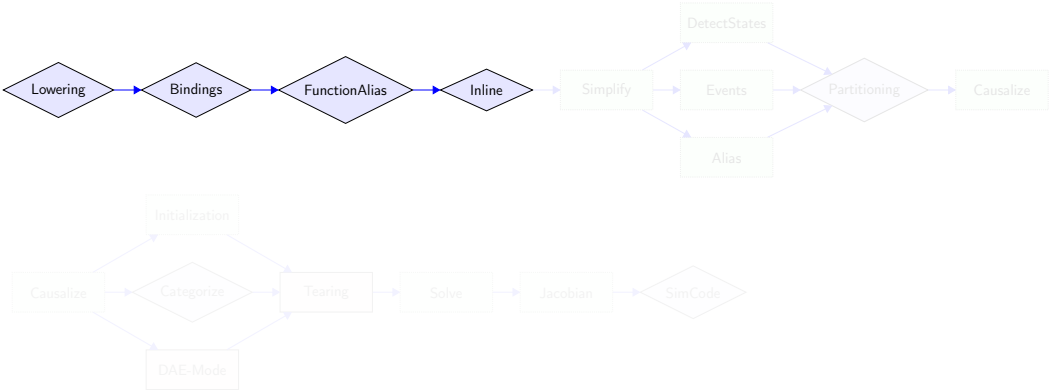
Status on Array-Handling



Finished | Core Finished | Work in Progress

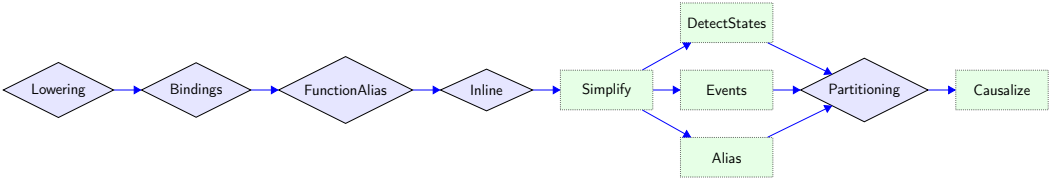
Backend Modules

Status on Array-Handling



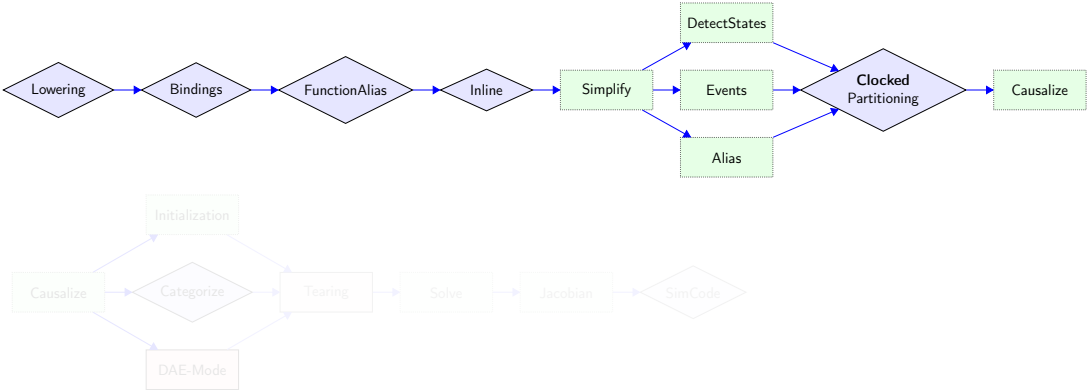
Backend Modules

Status on Array-Handling



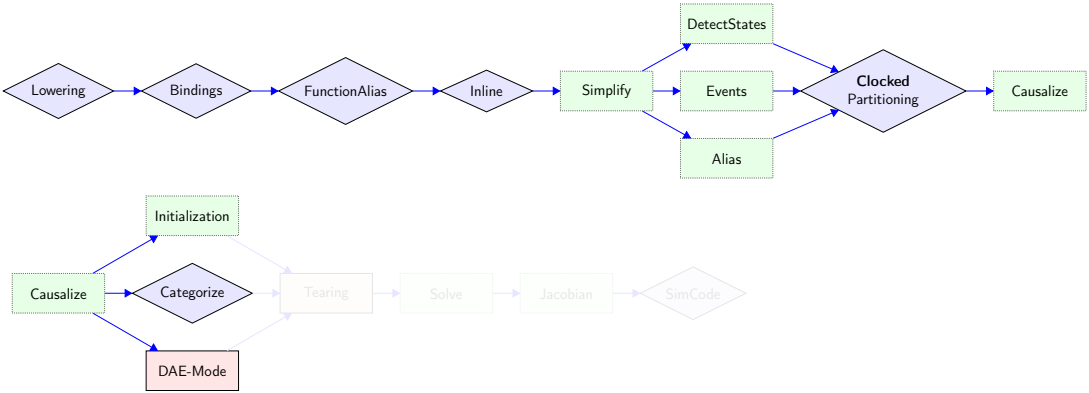
Backend Modules

Status on Array-Handling



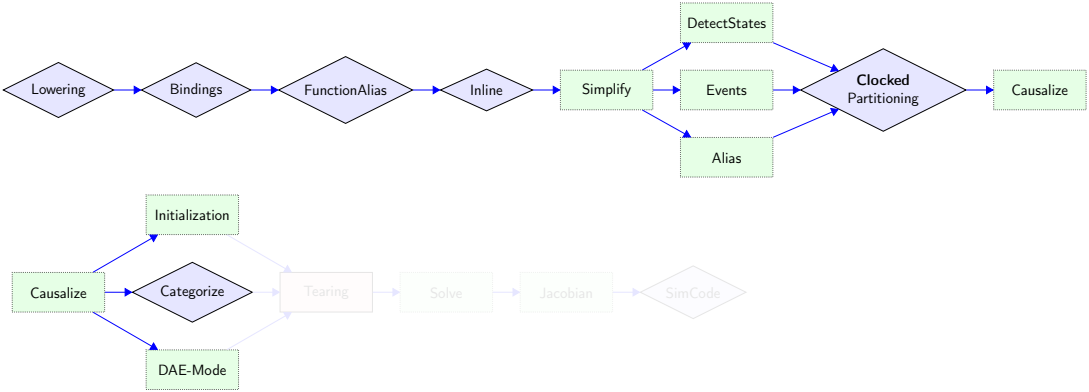
Backend Modules

Status on Array-Handling



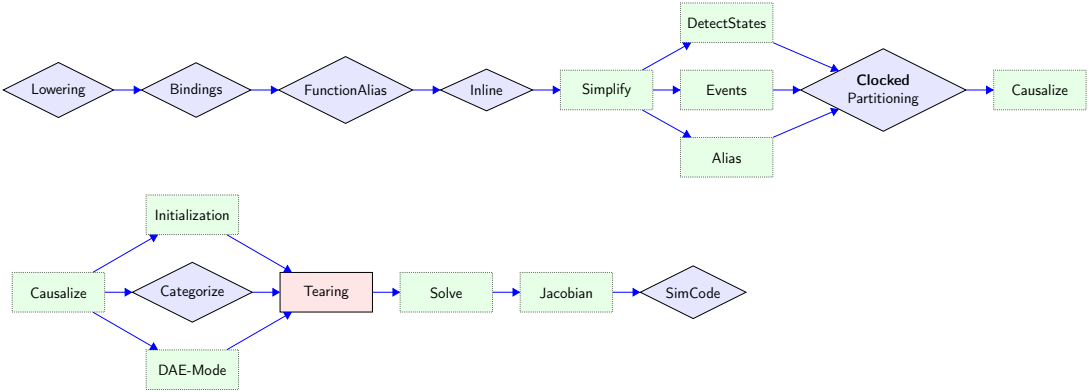
Backend Modules

Status on Array-Handling



Backend Modules

Status on Array-Handling



Section 2

Implicit Algorithms

Implicit Algorithms

Problem Definition

Algorithms

At first glance, algorithms in Modelica appear straightforward: variables on the left-hand side of statements are designated as outputs, while those on the right hand side serve as inputs.

Problem

However, this approach falls short in accommodating the flexibility inherent in the Modelica language specification, which permits algorithms to be solved implicitly for input variables.

Naive Solution

Considering the full algorithm as implicit in all solved variables whenever it has to be solved for any input.

Implicit Algorithms

Problem Definition

Algorithms

At first glance, algorithms in Modelica appear straightforward: variables on the left-hand side of statements are designated as outputs, while those on the right hand side serve as inputs.

Problem

However, this approach falls short in accommodating the flexibility inherent in the Modelica language specification, which permits algorithms to be solved implicitly for input variables.

Naive Solution

Considering the full algorithm as implicit in all solved variables whenever it has to be solved for any input.

Implicit Algorithms

Problem Definition

Algorithms

At first glance, algorithms in Modelica appear straightforward: variables on the left-hand side of statements are designated as outputs, while those on the right hand side serve as inputs.

Problem

However, this approach falls short in accommodating the flexibility inherent in the Modelica language specification, which permits algorithms to be solved implicitly for input variables.

Naive Solution

Considering the full algorithm as implicit in all solved variables whenever it has to be solved for any input.

Implicit Algorithms

Inputs, Outputs and Arrays

Natural Outputs

Let L denote the set of all component references assigned on the left hand side within an algorithm section. According to the Modelica language specification, if any $l \in L$ is a subscripted component reference referring to an array variable, then the entire array is considered assigned by the algorithm.

$$O = \{o | \exists l \in L \text{ where } l \text{ references } o\}. \tag{1}$$

Natural Inputs

Let R denote the set of all component references accessed on the right hand side within an algorithm section. Since output variables may also appear on the right hand side, they must be excluded when identifying the set of input variables.

$$I = \{i | \exists r \in R \text{ where } r \text{ references } i\} \setminus O \tag{2}$$

Implicit Algorithms

Inputs, Outputs and Arrays

Natural Outputs

Let L denote the set of all component references assigned on the left hand side within an algorithm section. According to the Modelica language specification, if any $l \in L$ is a subscripted component reference referring to an array variable, then the entire array is considered assigned by the algorithm.

$$O = \{o | \exists l \in L \text{ where } l \text{ references } o\}. \tag{1}$$

Natural Inputs

Let R denote the set of all component references accessed on the right hand side within an algorithm section. Since output variables may also appear on the right hand side, they must be excluded when identifying the set of input variables.

$$I = \{i | \exists r \in R \text{ where } r \text{ references } i\} \setminus O \tag{2}$$

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

Possible Matching Result

During the matching process, exactly $|O|$ scalar variables Y are matched to the algorithm, these may include variables from the input set I.

1. $O_Y = O \cap Y$ The subset of output variables actually matched to the algorithm.
2. $O_U = O \setminus Y$ The subset of output variables that are *not* matched to this algorithm.
3. $I_Y = I \cap Y$ The subset of input variables that are unexpectedly matched to this algorithm.
4. $I_U = I \setminus Y$ The subset of input variables that are not matched to the algorithm.

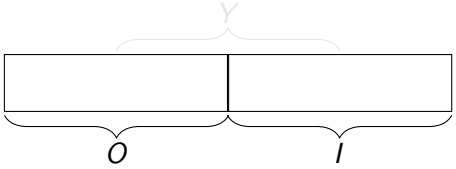
Actual Matching Result

The algorithm must be structurally solved for $Y = O_Y \cup I_Y$, while the algorithm body is symbolically structured with the assumption that it is solving for all declared outputs $O = O_Y \cup O_U$.

Implicit Algorithms

Parsing any Matching Result

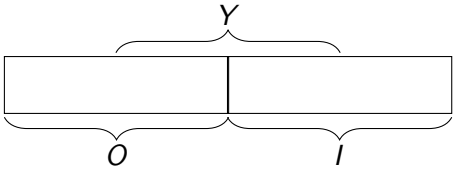
all variables:



Implicit Algorithms

Parsing any Matching Result

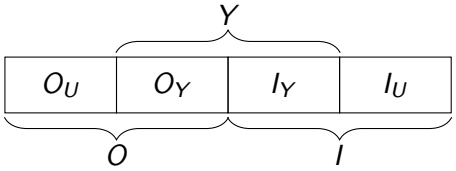
all variables:



Implicit Algorithms

Parsing any Matching Result

all variables:



Implicit Algorithms

Parsing any Matching Result

Default Case

$$Y = O_Y = O \quad \wedge \quad I_Y = \emptyset$$

All natural outputs are the actual outputs. No special handling needed.

Inverted Case

$$Y = I_Y \subset I \quad \wedge \quad O_Y = \emptyset$$

The matched variables are a subset of the natural inputs. If the algorithm is symbolically invertible a new explicit representation can be found. No implicit handling needed.

Implicit Algorithms

Parsing any Matching Result

Default Case

$$Y = O_Y = O \quad \wedge \quad I_Y = \emptyset$$

All natural outputs are the actual outputs. No special handling needed.

Inverted Case

$$Y = I_Y \subset I \quad \wedge \quad O_Y = \emptyset$$

The matched variables are a subset of the natural inputs. If the algorithm is symbolically invertible a new explicit representation can be found. No implicit handling needed.

Implicit Algorithms

Parsing any Matching Result

Implicit Case

$$Y = O_Y \cup I_Y \quad \wedge \quad O_Y \neq \emptyset \quad \wedge \quad I_Y \neq \emptyset$$

The matched variables consist of natural outputs and natural inputs.
General solution requires implicit handling using a nonlinear iterative solver. The core idea is to prepare the algorithm as a torn system, allowing it to be solved sequentially as an inner equation within an algebraic loop.

Problems to Solve

1. If the algorithm is an inner equation, what are the residual equations?
2. What are the iteration variables and what are the inner variables?
3. What happens if an array variable is only partially matched?

Implicit Algorithms

Parsing any Matching Result

Implicit Case

$$Y = O_Y \cup I_Y \quad \wedge \quad O_Y \neq \emptyset \quad \wedge \quad I_Y \neq \emptyset$$

The matched variables consist of natural outputs and natural inputs.
General solution requires implicit handling using a nonlinear iterative solver. The core idea is to prepare the algorithm as a torn system, allowing it to be solved sequentially as an inner equation within an algebraic loop.

Problems to Solve

1. If the algorithm is an inner equation, what are the residual equations?
2. What are the iteration variables and what are the inner variables?
3. What happens if an array variable is only partially matched?

Implicit Algorithms

Parsing any Matching Result

Implicit Case

$$Y = O_Y \cup I_Y \quad \wedge \quad O_Y \neq \emptyset \quad \wedge \quad I_Y \neq \emptyset$$

The matched variables consist of natural outputs and natural inputs.
General solution requires implicit handling using a nonlinear iterative solver. The core idea is to prepare the algorithm as a torn system, allowing it to be solved sequentially as an inner equation within an algebraic loop.

Problems to Solve

1. If the algorithm is an inner equation, what are the residual equations?
2. What are the iteration variables and what are the inner variables?
3. What happens if an array variable is only partially matched?

Implicit Algorithms

Parsing any Matching Result

Implicit Case

$$Y = O_Y \cup I_Y \quad \wedge \quad O_Y \neq \emptyset \quad \wedge \quad I_Y \neq \emptyset$$

The matched variables consist of natural outputs and natural inputs.
General solution requires implicit handling using a nonlinear iterative solver. The core idea is to prepare the algorithm as a torn system, allowing it to be solved sequentially as an inner equation within an algebraic loop.

Problems to Solve

1. If the algorithm is an inner equation, what are the residual equations?
2. What are the iteration variables and what are the inner variables?
3. What happens if an array variable is only partially matched?

Implicit Algorithms

If the algorithm is an inner equation, what are the residual equations?

Structural Facts

If the algorithm is solved sequentially as in inner equation, all natural output variables O of the algorithm must be computed within it—even if some of them, specifically $O_U \in O$, are already computed elsewhere in the model.

Local Artificial Inner Variables

To reconcile this, an artificial variable $\tilde{o}_U \in \tilde{O}_U$ is introduced for each $o_U \in O_U$, effectively creating local duplicates of these outputs.

Local Artificial Residual Equations

The algorithm is then modified to compute these artificial variables, while the original values of O_U, determined elsewhere, are used to define residual equations of the form:

$$0 = o_U - \tilde{o}_U \quad \forall \tilde{o}_U \in \tilde{O}_U \tag{3}$$

Implicit Algorithms

If the algorithm is an inner equation, what are the residual equations?

Structural Facts

If the algorithm is solved sequentially as in inner equation, all natural output variables O of the algorithm must be computed within it—even if some of them, specifically $O_U \in O$, are already computed elsewhere in the model.

Local Artificial Inner Variables

To reconcile this, an artificial variable $\tilde{o}_U \in \tilde{O}_U$ is introduced for each $o_U \in O_U$, effectively creating local duplicates of these outputs.

Local Artificial Residual Equations

The algorithm is then modified to compute these artificial variables, while the original values of O_U, determined elsewhere, are used to define residual equations of the form:

$$0 = o_U - \tilde{o}_U \quad \forall \tilde{o}_U \in \tilde{O}_U \tag{3}$$

Implicit Algorithms

If the algorithm is an inner equation, what are the residual equations?

Structural Facts

If the algorithm is solved sequentially as in inner equation, all natural output variables O of the algorithm must be computed within it—even if some of them, specifically $O_U \in O$, are already computed elsewhere in the model.

Local Artificial Inner Variables

To reconcile this, an artificial variable $\tilde{o}_U \in \tilde{O}_U$ is introduced for each $o_U \in O_U$, effectively creating local duplicates of these outputs.

Local Artificial Residual Equations

The algorithm is then modified to compute these artificial variables, while the original values of O_U, determined elsewhere, are used to define residual equations of the form:

$$0 = o_U - \tilde{o}_U \quad \forall \tilde{o}_U \in \tilde{O}_U \tag{3}$$

Implicit Algorithms

What are the iteration variables and what are the inner variables?

Inner Variables

All natural outputs O are considered inner variables (partially aliased).

Local Artificial Inner Variables Artificial variables $\tilde{O}_U$.

Local Inner Variables All natural actual outputs O_Y.

Iteration Variables

All natural inputs that are matched as an output (I_Y) are iteration variables.

Locally Known Variables

All natural inputs that are matched as an input (I_U) are computed outside the algorithm and just act as inputs. They have no further meaning and are considered known for solving the algebraic loop.

Implicit Algorithms

What are the iteration variables and what are the inner variables?

Inner Variables

All natural outputs O are considered inner variables (partially aliased).

Local Artificial Inner Variables Artificial variables $\tilde{O}_U$.

Local Inner Variables All natural actual outputs O_Y.

Iteration Variables

All natural inputs that are matched as an output (I_Y) are iteration variables.

Locally Known Variables

All natural inputs that are matched as an input (I_U) are computed outside the algorithm and just act as inputs. They have no further meaning and are considered known for solving the algebraic loop.

Implicit Algorithms

What are the iteration variables and what are the inner variables?

Inner Variables

All natural outputs O are considered inner variables (partially aliased).

Local Artificial Inner Variables Artificial variables $\tilde{O}_U$.

Local Inner Variables All natural actual outputs O_Y.

Iteration Variables

All natural inputs that are matched as an output (I_Y) are iteration variables.

Locally Known Variables

All natural inputs that are matched as an input (I_U) are computed outside the algorithm and just act as inputs. They have no further meaning and are considered known for solving the algebraic loop.

Implicit Algorithms

What happens if an array variable is only partially matched?

Partially Unmatched Variables in O_U

If an array variable appears only partially in the set O_U, the entire array must still be replaced by an alias. As a result, the residual equation must also be generated for the entire array, even if only a subset of its elements is actually unmatched.

Partially Matched Variables in O_Y

Those elements of the array $\tilde{O}_Y$ that are replaced—even if originally part of the matched output set O_Y—must be included in the set of iteration variables.

Partially Matched Variables in I

These variables do not require further handling as array handling only influences natural outputs O as they are considered to be fully solved by the algorithm body by modelica language specification.

Implicit Algorithms

What happens if an array variable is only partially matched?

Partially Unmatched Variables in O_U

If an array variable appears only partially in the set O_U, the entire array must still be replaced by an alias. As a result, the residual equation must also be generated for the entire array, even if only a subset of its elements is actually unmatched.

Partially Matched Variables in O_Y

Those elements of the array $\tilde{O}_Y$ that are replaced—even if originally part of the matched output set O_Y—must be included in the set of iteration variables.

Partially Matched Variables in I

These variables do not require further handling as array handling only influences natural outputs O as they are considered to be fully solved by the algorithm body by modelica language specification.

Implicit Algorithms

What happens if an array variable is only partially matched?

Partially Unmatched Variables in O_U

If an array variable appears only partially in the set O_U, the entire array must still be replaced by an alias. As a result, the residual equation must also be generated for the entire array, even if only a subset of its elements is actually unmatched.

Partially Matched Variables in O_Y

Those elements of the array $\tilde{O}_Y$ that are replaced—even if originally part of the matched output set O_Y—must be included in the set of iteration variables.

Partially Matched Variables in I

These variables do not require further handling as array handling only influences natural outputs O as they are considered to be fully solved by the algorithm body by modelica language specification.

Implicit Algorithms

Final Structure

	equations	variables
inner	original algorithm	matched output variables O_Y alias unmatched output variables $\tilde{O}_U$
iteration	artifical residual equations $0 = o_U - \tilde{o}_U$	matched input variables I_Y alias matched array variables $\tilde{O}_Y$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a$;	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a$;	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm ($a \rightarrow \$TMP.a$)	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm $(a \rightarrow \$TMP.a)$	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Example

```
model Example
  Real u, a;
  Real b[1000];
algorithm
  b[1] := 2*u;
  for i in 2:1000 loop
    b[i] := b[1] + b[i-1];
  end for;
  a := sqrt(b[500]^2 +
    b[1000]^2);
equation
  a = 10;
end Example;
```

1. Naturals: $O = \{a, b\} \quad \wedge \quad I = \{u\}$
2. Matched: $Y = \{u, b\}$
3. $O_Y = O \cap Y = \{b\}$
4. $O_U = O \setminus Y = \{a\}$
5. $I_Y = I \cap Y = \{u\}$
6. $I_U = I \setminus Y = \emptyset$

	equations	variables
inner	algorithm $(a \rightarrow \$TMP.a)$	$\{b, \$TMP.a\}$
iteration	$0 = a - \$TMP.a;$	$\{u\}$

Implicit Algorithms

Summary

Naive Approach

Wrap the algorithm in a function and solve implicitely.

```
model Example_f
  Real u, a;
  Real b[1000];
  function f_algorithm
    input Real u;
    output Real b[1000], a;
  algorithm
    a := 0;
    b := zeros(1000);
    b[1] := 2*u;
    for i in 2:1000 loop
      b[i] := b[1] + b[i-1];
    end for;
    a := sqrt(b[500]^2 + b[1000]^2);
  end f_algorithm;
equation
  (b, a) = f_algorithm(u);
  a = 10;
end Example_f;
```

Benefits of the presented Method

The number of iteration variables could be drastically reduced to matched input variables and the number of alias matched array variables $I_Y \cup \tilde{O}_Y$ wheras the naive approach would iterate over Y.

Applied to the Example

Naive: 1001 Iteration Variables.
1001×1001 Jacobian

Smart: 1 Iteration Variable.
1×1 Jacobian

Implicit Algorithms

Summary

Naive Approach

Wrap the algorithm in a function and solve implicitely.

```
model Example_f
  Real u, a;
  Real b[1000];
  function f_algorithm
    input Real u;
    output Real b[1000], a;
  algorithm
    a := 0;
    b := zeros(1000);
    b[1] := 2*u;
    for i in 2:1000 loop
      b[i] := b[1] + b[i-1];
    end for;
    a := sqrt(b[500]^2 + b[1000]^2);
  end f_algorithm;
equation
  (b, a) = f_algorithm(u);
  a = 10;
end Example_f;
```

Benefits of the presented Method

The number of iteration variables could be drastically reduced to matched input variables and the number of alias matched array variables $I_Y \cup \tilde{O}_Y$ wheras the naive approach would iterate over Y.

Applied to the Example

Naive: 1001 Iteration Variables.
1001×1001 Jacobian

Smart: 1 Iteration Variable.
1×1 Jacobian

Implicit Algorithms

Summary

Naive Approach

Wrap the algorithm in a function and solve implicitely.

```
model Example_f
  Real u, a;
  Real b[1000];
  function f_algorithm
    input Real u;
    output Real b[1000], a;
  algorithm
    a := 0;
    b := zeros(1000);
    b[1] := 2*u;
    for i in 2:1000 loop
      b[i] := b[1] + b[i-1];
    end for;
    a := sqrt(b[500]^2 + b[1000]^2);
  end f_algorithm;
equation
  (b, a) = f_algorithm(u);
  a = 10;
end Example_f;
```

Benefits of the presented Method

The number of iteration variables could be drastically reduced to matched input variables and the number of alias matched array variables $I_Y \cup \tilde{O}_Y$ wheras the naive approach would iterate over Y.

Applied to the Example

Naive: 1001 Iteration Variables.
1001×1001 Jacobian

Smart: 1 Iteration Variable.
1×1 Jacobian

Section 3

Summary

Summary

Recent Development

- Implicit Algorithms
- Static Index Reduction
- Better DAEMode Support
- Basic Clocked Support

Current Development

- Jacobian and sparsity updates
- Target code resizable support

Summary

Recent Development

- Implicit Algorithms
- Static Index Reduction
- Better DAEMode Support
- Basic Clocked Support

Current Development

- Jacobian and sparsity updates
- Target code resizable support