Vector Enumeration

Florian Heine, Nicolaus Jacobsen

Vector Enumeration[1]

Linear version of the Todd-Coxeter algorithm; by S.A. Linton

Input

For a field K and a finite set of symbols X let $A := K\langle X\rangle$ and $F := A^s$:

- $P := \langle X \mid R\rangle_K$ a finitely presented K-algebra
- $M := \langle e_1, ..., e_s \mid W\rangle$ a finitely presented right P-module

Output

- K-matrix representation of M as K-vectorspace
- monomial K-basis of M

---

[1]S.A. Linton, On vector enumeration, Linear Algebra and its Applications, Volume 192, 1993, Pages 235-248, ISSN 0024-3795

Interface[2]

```
julia> using AbstractAlgebra, VectorEnumeration

julia> A, (a, b, c) = free_associative_algebra(GF(7), ["a", "b", "c"])

julia> F = free_module(A, 1)

julia> R = [a^2 - 1, b^2 - 1, c^2 - 1, (a*b)^3 - 1, (a*c)^2 - 1, (b*c)^3 - 1]

julia> W = [F([a*b*c - 1])]
```

Dimension:

```
julia> dimension_qm(A, F, R, W)
6
```

[2] https://ktrompfl.github.io/VectorEnumeration.jl/dev/api/

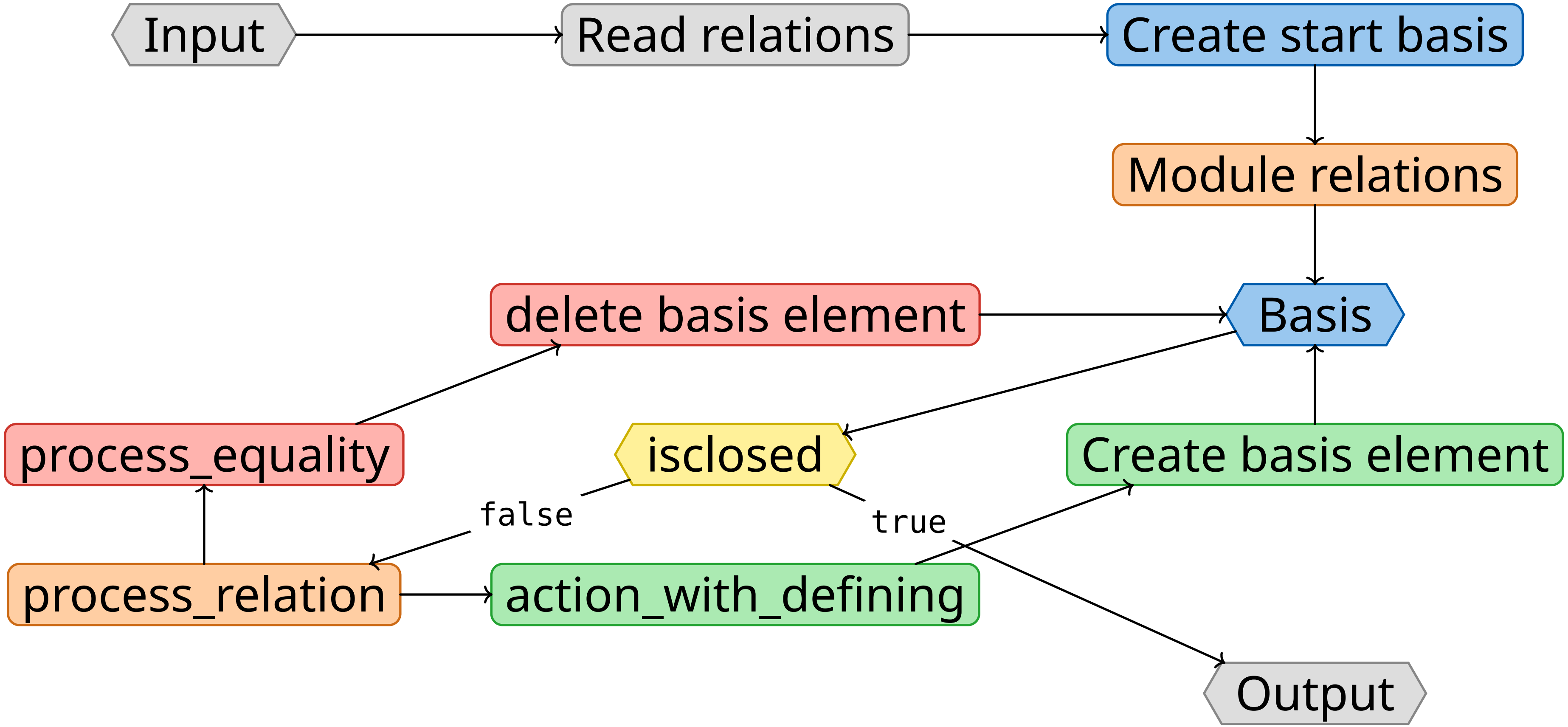
Matrices:

```
julia> Xᵃ, Xᵇ, Xᶜ = matrices_qm(Matrix, A, F, R, W)

julia> Xᵃ
6×6 Matrix{...}:
 0  1  0  0  0  0
 1  0  0  0  0  0
 0  0  0  1  0  0
 0  0  1  0  0  0
 0  0  0  0  0  1
 0  0  0  0  1  0

julia> Xᵇ
6×6 Matrix{...}:
 0  0  0  0  1  0
 0  0  1  0  0  0
 0  1  0  0  0  0
 0  0  0  0  0  1
 1  0  0  0  0  0
 0  0  0  1  0  0

julia> Xᶜ
6×6 Matrix{...}:
 0  0  1  0  0  0
 0  0  0  1  0  0
 1  0  0  0  0  0
 0  1  0  0  0  0
 0  0  0  0  0  1
 0  0  0  0  1  0
```

Monomial basis:

```
julia> base_qm(A, F, R, W)
6-element Vector{...}:
 (1), (a), (a*b), (a*b*a), (b), (a*b*a*b)
```

Main Program



Main Program

- relations r are initialized with a weight w_r, e.g. their degree
- basis elements b obtain the weight $w_b := w$ of the current iteration

```
w := 1
while w ≤ w_max:
    if isclosed(...):
        w := ∞
    for every relation r:
        for every basis element b:
            if w_b + w_r ≤ w and r was not applied to b yet:
                process_relation(b, r, w)
    w := w + 1
```

Basis Elements

Every basis element $b_i \in B = \{b_1, \dots, b_n\}$ saves the following data:

- $d_i \in \{\mathbf{true}, \mathbf{false}\}$: b_i deleted / undeleted
- $r_i \in K\{b_j \in B \mid j < i\} \cup \{\perp\}$: replacement, if b_i is deleted
- $p_i \in \langle X \rangle^s$: the monomial image of b_i
- $b_i[x] \in KB \cup \{\perp\}$: Action $b_i.x$ of $x \in X$ on b_i

$B^u := \{b_i \in B \mid d_i = \mathbf{false}\}$ – Set of undeleted basis elements

for $x \in X$, $b \in B$ s. th. $b[x] \neq \perp$ define $b.x = b[x]$

Algorithm

Consider the following input:

```
julia> A, (x,y) = free_associative_algebra(GF(3), [:x, :y])
julia> R = [x^2 + 2*x + 1,x*y - y*x, y^2 - 1]
julia> F = free_module(A, 2)
julia> W = [F([one(A),one(A)])]
```

- Read in relations:

  $\rightarrow$
  ```
  found inverse y to generator y
  AlgebraRelation(x^2 + 2*x + 2, Weight(3))
  BinomialRelation(1, x*y, 1, y*x, Weight(3))
  DefineRelation(x, Weight(3))
  DefineRelation(y, Weight(3))
  ```

- Create start basis $B = \{b_1, ..., b_s\}$ mit $p_i := e_i$:

  $\rightarrow$
  ```
  creating the start basis with 2 elements
  ```

Algorithm

- Process module relations:

  → processing submodule generator $e_1 + e_2$ at weight 1

- Process relations in the main loop:

  → processing fixing relation b_1.x^2 + 2*x + 2 = b_1 at weight 5
  processing binomial relation (1·b_1).x*y = (1·b_1).y*x at weight 5
  verifying b_1.x is defined at weight 5
  verifying b_1.y is defined at weight 5

- Create output:

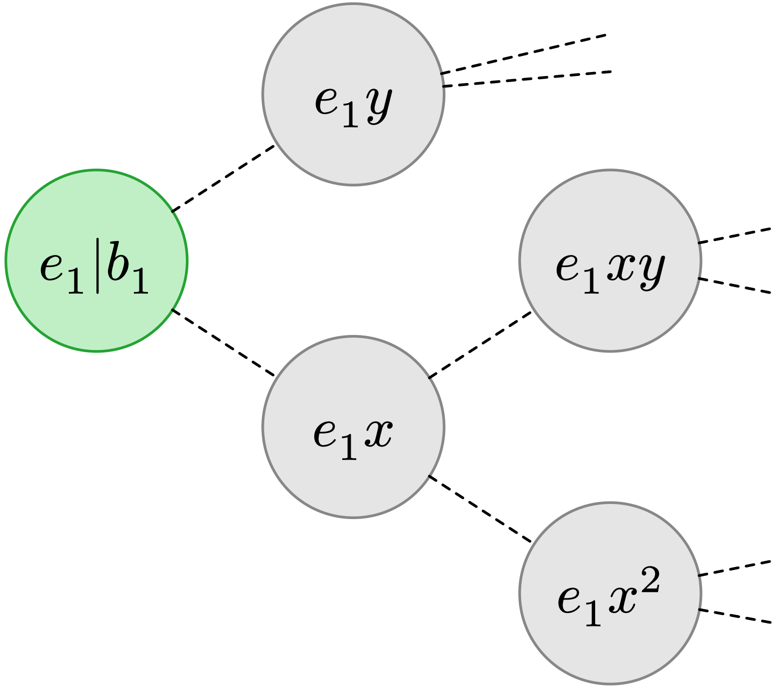
  → early closing at weight 10
  finished with dimension 4 after defining 10 basis elements
  collecting basis

How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x$, $e_1 x^2$:

- compute $b_1.x^2$:

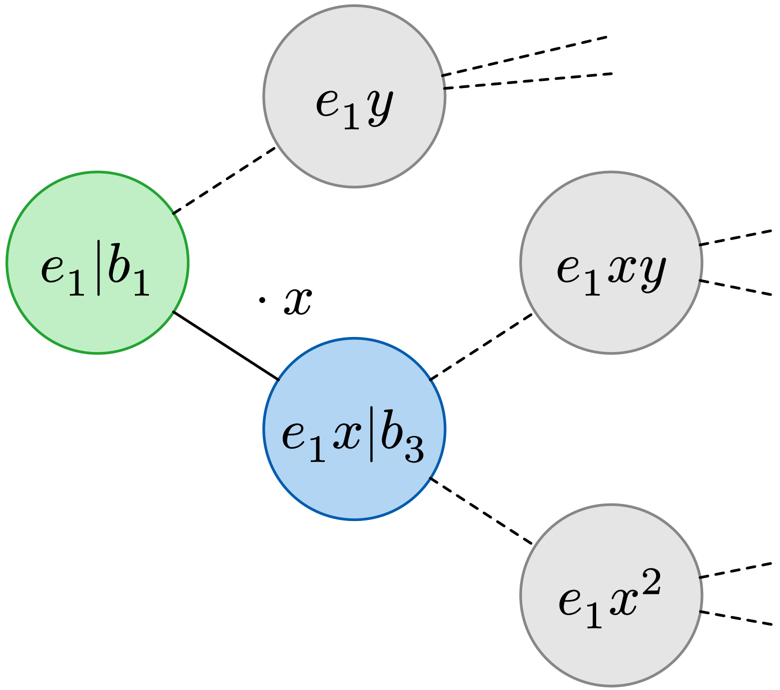


How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x$, $e_1 x^2$:

- compute $b_1.x^2$:
  - define $b_1[x] = b_1.x := b_3$
    $\rightarrow$ enumerate new basis element b_3



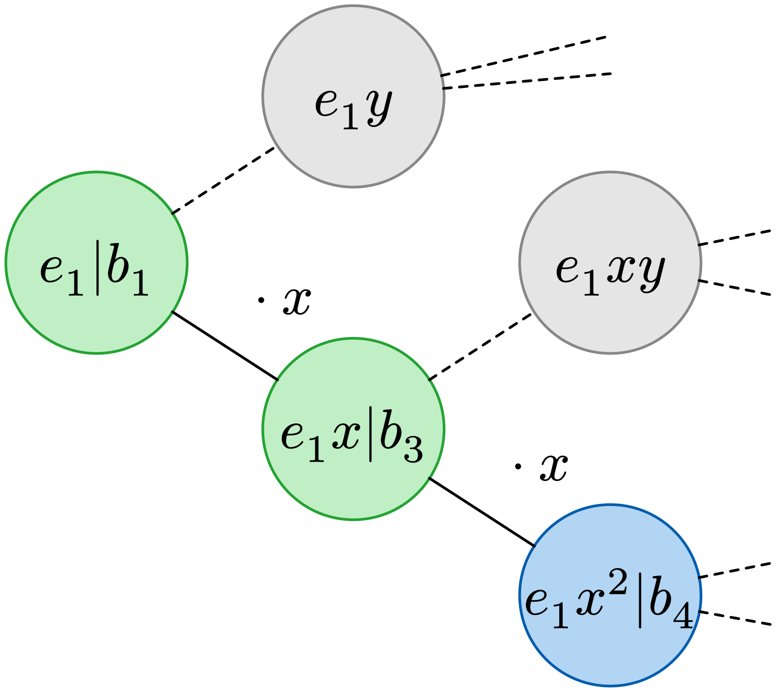
How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x, e_1 x^2$:

- compute $b_1.x^2$:
  - define $b_1[x] = b_1.x := b_3$
  - define $b_3[x] = b_3.x := b_4$

    $\to$ enumerate new basis element b_4



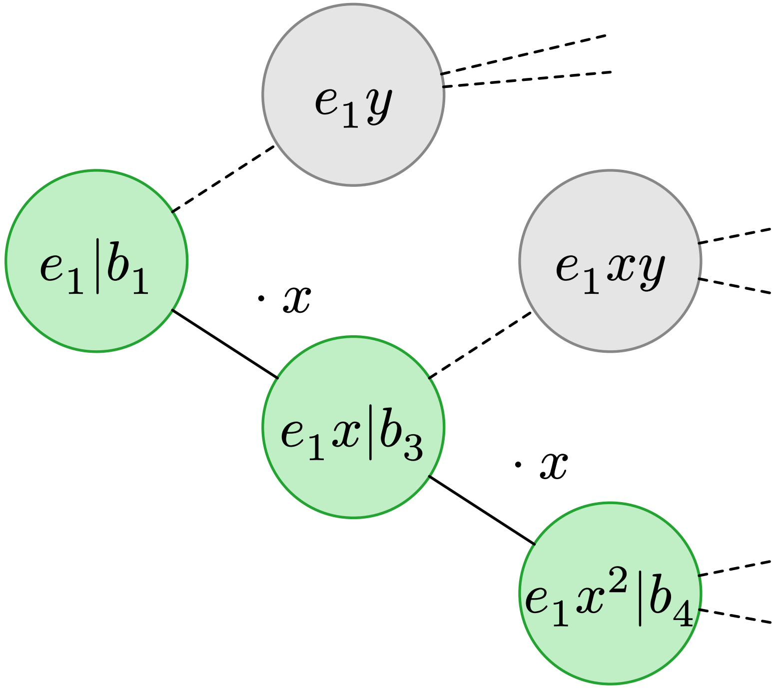
How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x, e_1 x^2$:

- compute $b_1.x^2$:
  - define $b_1[x] = b_1.x := b_3$
  - define $b_3[x] = b_3.x := b_4$

  $\Rightarrow b_1.x^2 = b_4$

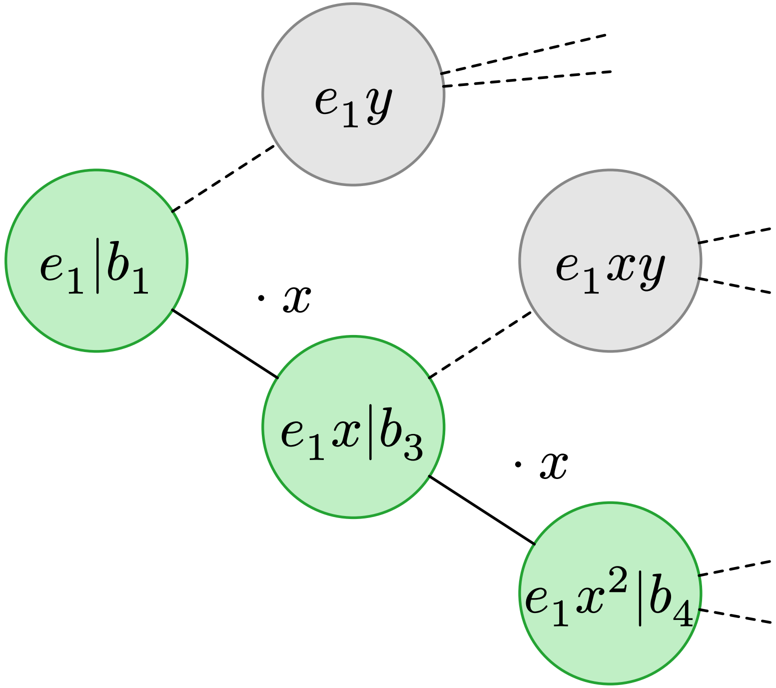


How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x, e_1 x^2$:

- compute $b_1.x^2$:
  $\Rightarrow b_1.x^2 = b_4$
- compute $b_1.2 \cdot x$:
  - $b_1.x = b_1[x] = b_3$
    $\Rightarrow b_1.2 \cdot x = 2b_3$



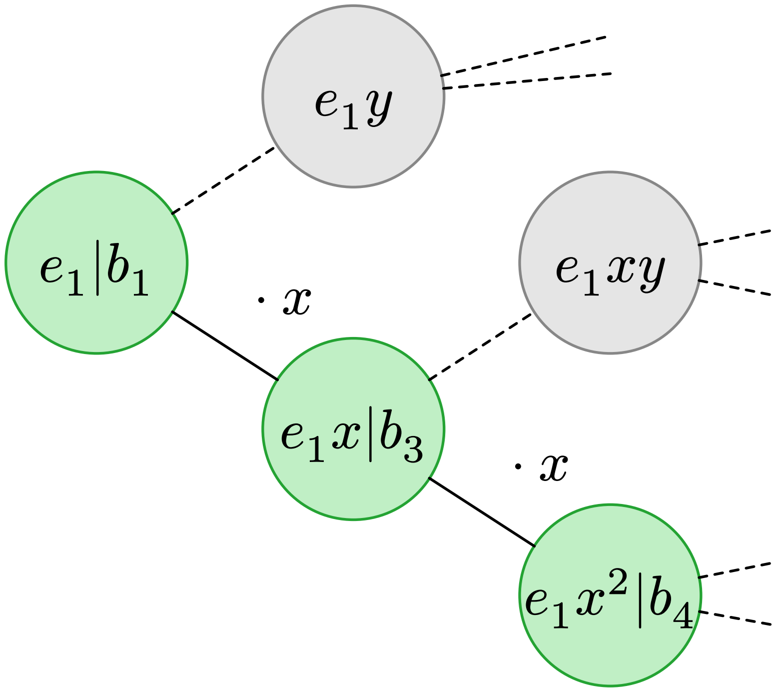
How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Compute $b_1.(x^2 + 2 \cdot x + 2)$ while defining new basis elements b_3, b_4, representative for discovered monomials $e_1 x, e_1 x^2$:

- compute $b_1.x^2$:
  $\Rightarrow b_1.x^2 = b_4$
- compute $b_1.2 \cdot x$:
  $\Rightarrow b_1.2 \cdot x = 2b_3$
- compute $b_1.2$:
  $\Rightarrow b_1.2 = 2b_1$

$\Rightarrow b_1.(x^2 + 2 \cdot x + 2) = b_4 + 2b_3 + 2b_1$

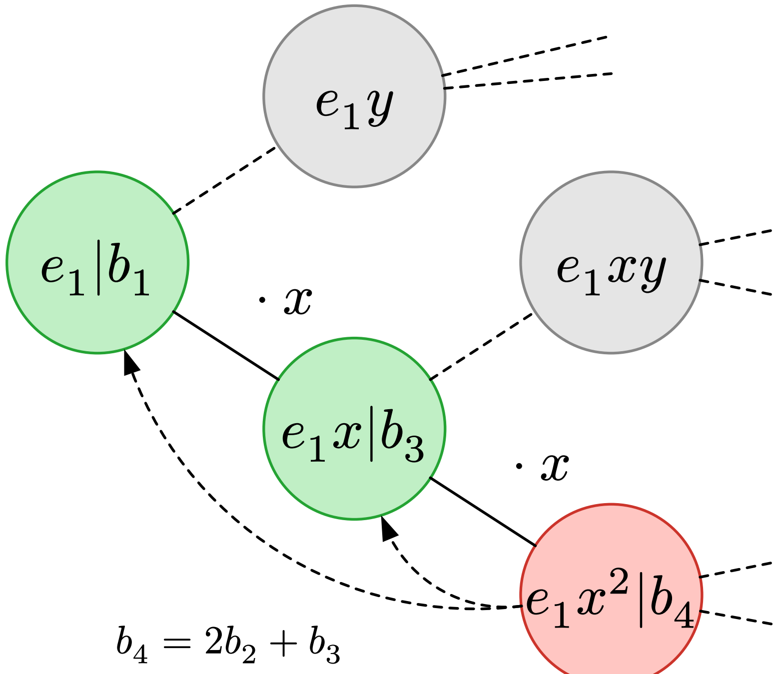


How to process relations?

```
processing fixing relation b₁.(x^2 + 2*x + 2) = b₁ at weight 5
```

Solve the equation $b_4 + 2b_3 + 2b_1 = b_1$ for b_4 to replace b_4:

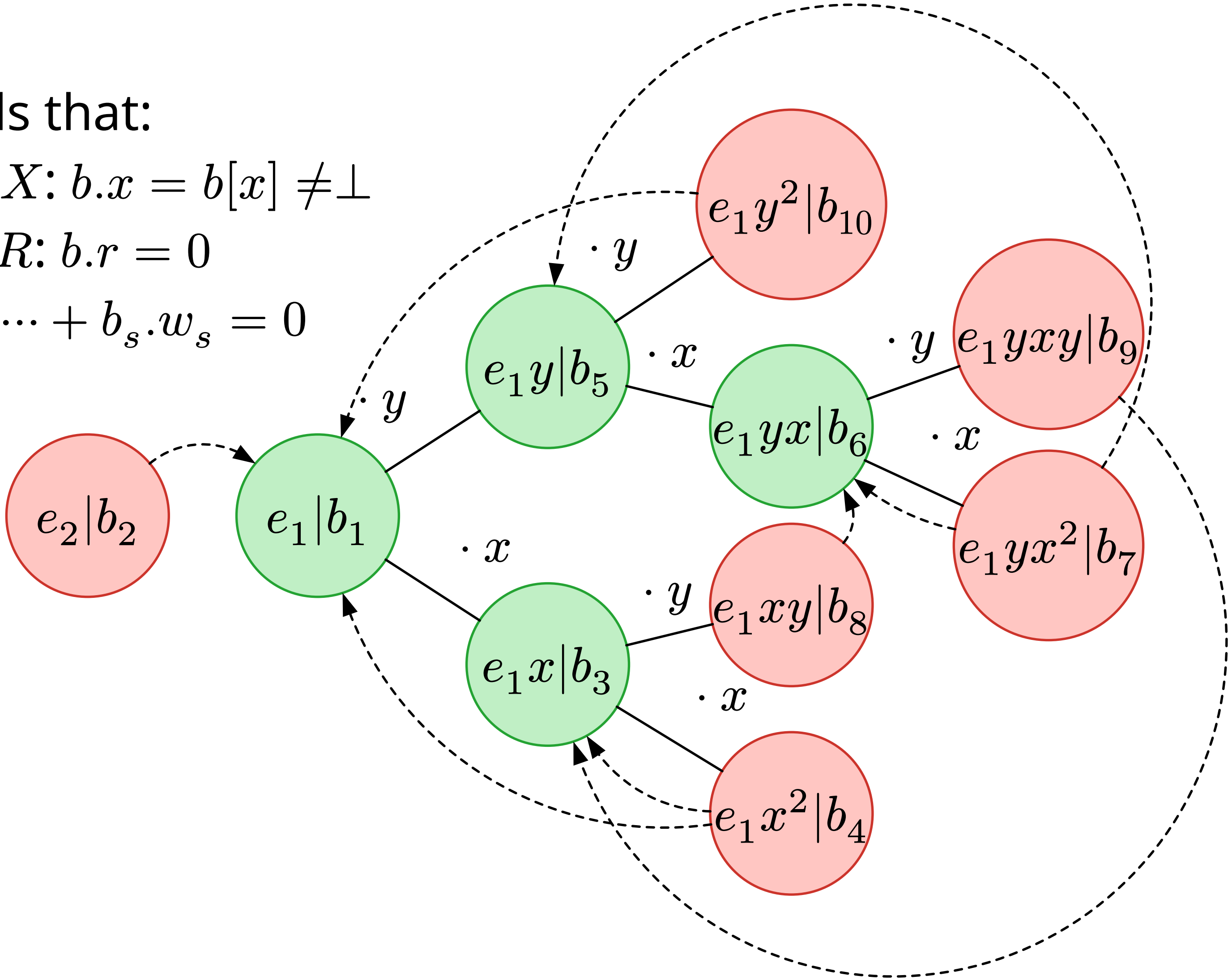
- solve the equation for b_4:
  $\Rightarrow b_4 = 2b_1 + b_3$
- replace and delete b_4:
  $\Rightarrow r_4 := 2b_1 + b_3$
  $\Rightarrow d_4 := \text{true}$



Termination

On terminitation it holds that:

- for all $b \in B^u$ and $x \in X$: $b.x = b[x] \neq \perp$
- for all $b \in B^u$ and $r \in R$: $b.r = 0$
- for all $w \in W$: $b_1.w_1 + \cdots + b_s.w_s = 0$



Benchmarks

Group[3]	Order	VectorEnumeration.jl		gap-packages/ve	
		Normal	Lookahead	Normal	Lookahead
$M_{11}^{(1)}$	7920	0.52 s	0.51 s	0.28 s	0.26 s
$M_{11}^{(2)}$	7920	0.51 s	0.47 s	0.38 s	0.31 s
$\mathrm{PSL}_3(4)$	20160	2.34 s	2.52 s	0.51 s	0.43 s
Neu	40320	101.34 s	69.07 s	61.15 s	31.96 s
$\mathrm{Weyl}\ B_6$	46080	1.35 s	2.85 s	0.64 s	0.81 s

[3] Cannon, John J., et al. “Implementation and Analysis of the Todd-Coxeter Algorithm.” Mathematics of Computation, vol. 27, no. 123, 1973, pp. 463–90.