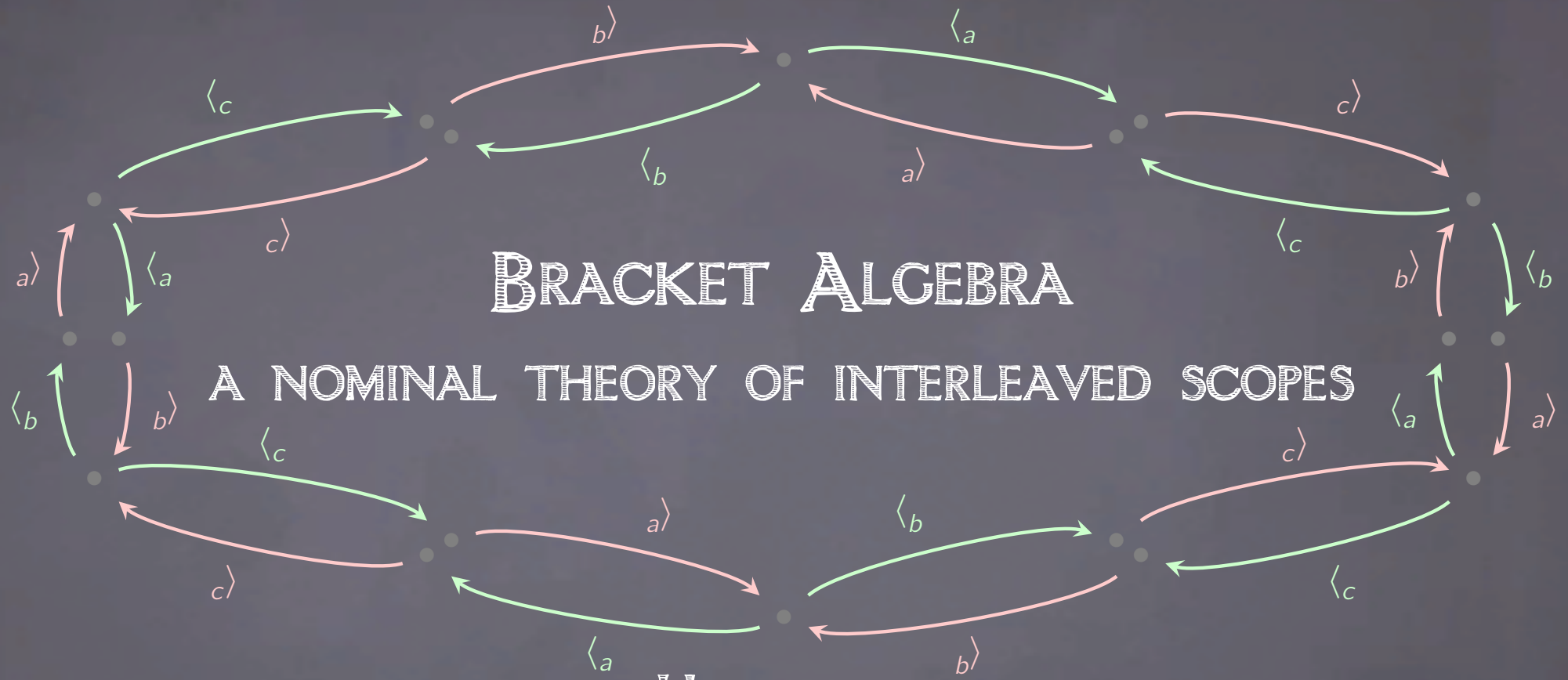


Bracket Algebra

a nominal theory of interleaved scopes

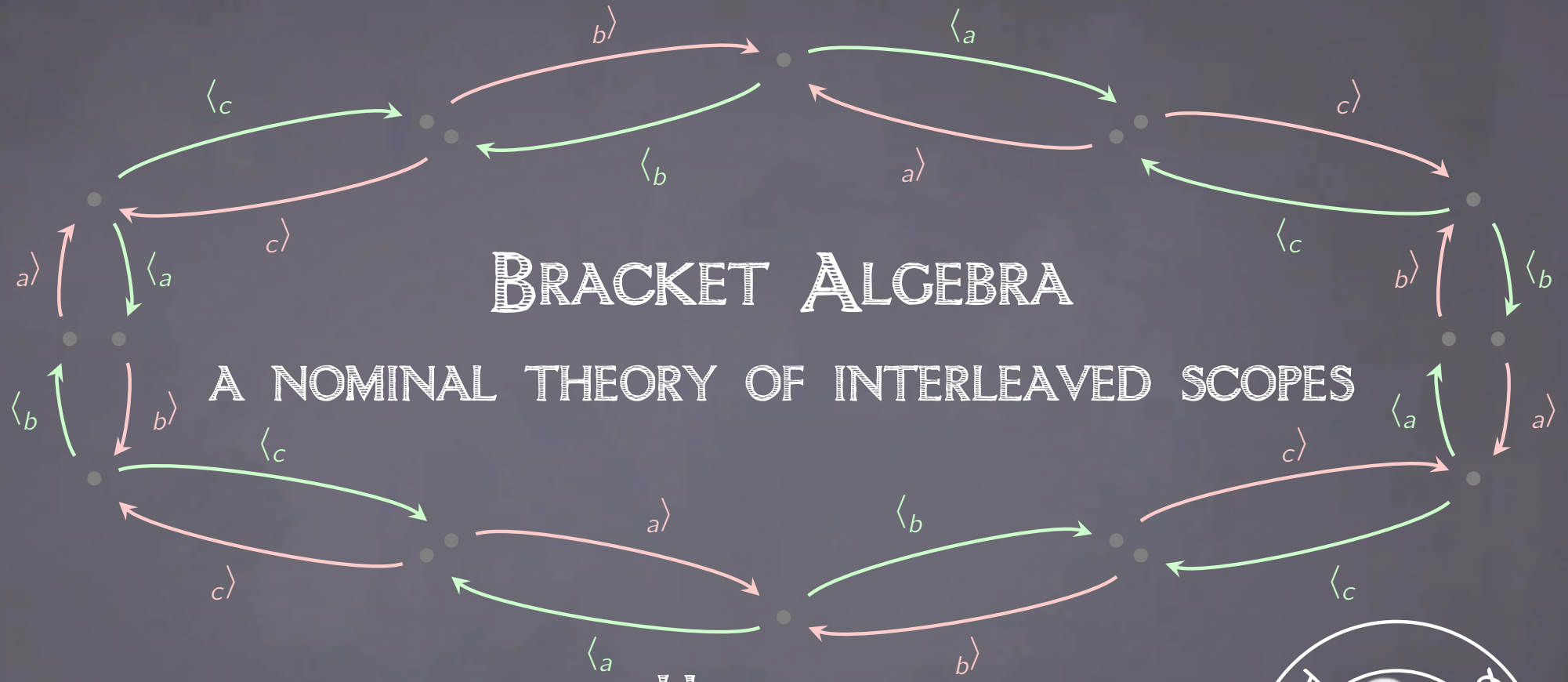
Highlights

September 18–21, 2018

Paul Brunet[1], Alexandra Silva[1], Daniela Petrişan[2]
[1]University College London, [2]Université Paris Diderot

UCL

université PARIS DIDEROT



Bracket Algebra

a nominal theory of interleaved scopes

Highlights

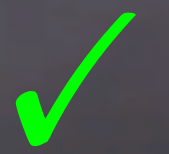
September 18–21, 2018

Paul Brunet[1], Alexandra Silva[1], Daniela Petrişan[2]
[1]University College London, [2]Université Paris Diderot

Kleene Alg. for program equivalence

```
int i;
int j;
j:=x;
x:=y;
x++;
for(i=1,i<=10,i++) x++;
free(i);
y:=j;
free(j);
```

```
int i;
int j;
j:=x;
x:=y;
for(i=1,i<=10,i++) x++;
x++;
free(i);
y:=j;
free(j);
```

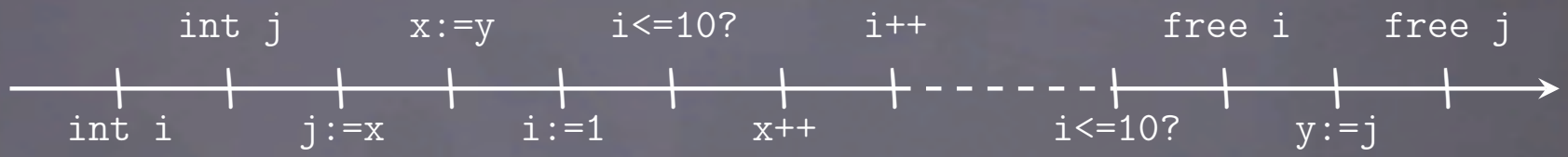


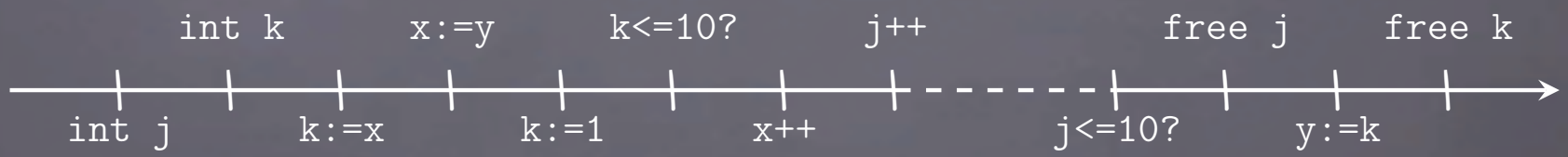
Kleene Alg. for program equivalence

```
int i;
int j;
j:=x;
x:=y;
x++;
for(i=1,i<=10,i++) x++;
free(i);
y:=j;
free(j);
```

```
int j;
int k;
k:=x;
x:=y;
for(j=1,j<=10,j++) x++;
x++;
free(j);
y:=k;
free(k);
```

α-EQUIVALENCE W. INTERLEAVED SCOPES





α-EQUIVALENCE W. INTERLEAVED SCOPES

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

Anatomy of a trace

$$\langle_a \quad \langle_b \quad x \quad y \quad {}_a\rangle \quad z \quad {}_b\rangle$$

Words

- set of names $\mathbb{A}$, nominal alphabet $\mathbb{X}$ of instructions
- words built from:
  - instructions x from $\mathbb{X}$
  - resource creation $\langle_a$, $(a \in \mathbb{A})$
  - resource destruction ${}_a\rangle$, $(a \in \mathbb{A})$.

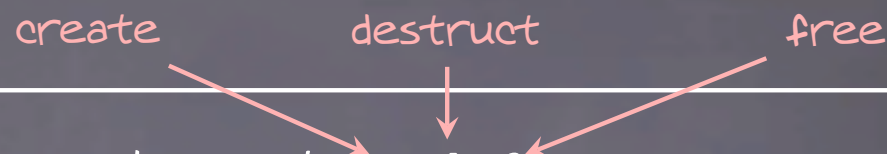
1) swap:

$$\langle_a\, [a := x][x := y][y := a]\, {}_a\rangle$$

2) scopes:

$${}_a\rangle\langle_c\, c\langle_a\, a\, {}_c\rangle\langle_b\, a\, {}_a\rangle\, a$$

Binding powers



Binding Monoid

- generated by three elements $\mathtt{c}, \mathtt{d}, \mathtt{f}$;
- quotiented by:

$$\mathtt{c} \cdot \mathtt{d} = 1 \qquad \mathtt{f} \cdot \mathtt{f} = \mathtt{f}$$
$$\mathtt{c} \cdot \mathtt{f} = \mathtt{c} \qquad \mathtt{f} \cdot \mathtt{d} = \mathtt{d}$$

Gabbay, Ghica & Petrişan, "Leaving the Nest", 2015

Binding powers

create destruct free

Binding Monoid

- Generated by three elements $\mathtt{c}, \mathtt{d}, \mathtt{f}$;
- Quotiented by:

$$\mathtt{c} \cdot \mathtt{d} = 1 \qquad \mathtt{f} \cdot \mathtt{f} = \mathtt{f}$$
$$\mathtt{c} \cdot \mathtt{f} = \mathtt{c} \qquad \mathtt{f} \cdot \mathtt{d} = \mathtt{d}$$

Gabbay, Ghica & Petrişan, "Leaving the Nest", 2015

Binding power of word w with respect to name a

$$w = \quad {}_{a}\rangle \quad \langle_{c} \quad c \quad \langle_{a} \quad a \quad {}_{c}\rangle \quad \langle_{b} \quad a \quad {}_{a}\rangle \quad a$$

Binding powers

create destruct free

Binding Monoid

- Generated by three elements $\mathtt{c}, \mathtt{d}, \mathtt{f}$;
- Quotiented by:

$$\mathtt{c} \cdot \mathtt{d} = 1 \qquad \mathtt{f} \cdot \mathtt{f} = \mathtt{f}$$
$$\mathtt{c} \cdot \mathtt{f} = \mathtt{c} \qquad \mathtt{f} \cdot \mathtt{d} = \mathtt{d}$$

Gabbay, Ghica & Petrişan, "Leaving the Nest", 2015

Binding power of w (word) with respect to a (name)

$$w = \underset{\mathtt{d}}{\langle_a} \quad \underset{1}{\langle_c} \quad \underset{1}{c} \quad \underset{\mathtt{c}}{\langle_a} \quad \underset{\mathtt{f}}{a} \quad \underset{1}{c\rangle} \quad \underset{1}{\langle_b} \quad \underset{\mathtt{f}}{a} \quad \underset{\mathtt{d}}{a\rangle} \quad \underset{\mathtt{f}}{a}$$

Binding powers

Binding Monoid

create destruct free

- Generated by three elements $\mathtt{c}, \mathtt{d}, \mathtt{f}$;
- Quotiented by:

$$\mathtt{c}\cdot\mathtt{d} = 1 \qquad \mathtt{f}\cdot\mathtt{f} = \mathtt{f}$$
$$\mathtt{c}\cdot\mathtt{f} = \mathtt{c} \qquad \mathtt{f}\cdot\mathtt{d} = \mathtt{d}$$

Gabbay, Ghica & Petrişan, "Leaving the Nest", 2015

Binding power of word w with respect to name a

$$w = \underset{\mathtt{d}}{\langle_a} \quad \underset{1}{\langle_c} \quad \underset{1}{c} \quad \underset{c}{\langle_a} \quad \underset{\mathtt{f}}{a} \quad \underset{1}{c\rangle} \quad \underset{1}{\langle_b} \quad \underset{\mathtt{f}}{a} \quad \underset{\mathtt{d}}{a\rangle} \quad \underset{\mathtt{f}}{a}$$

$$\mathtt{d}\cdot\mathtt{c}\cdot\mathtt{f}\cdot\mathtt{f}\cdot\mathtt{d}\cdot\mathtt{f} = \mathtt{df} =: \mathcal{F}_a(w)$$

Alpha-equivalence

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

Alpha-equivalence

$$\boxed{a \# u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \# u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

Alpha-equivalence

$$\boxed{a \# u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \# u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

$$\langle_a\, a\, {}_a\rangle =_\alpha \langle_b\, b\, {}_b\rangle$$

Alpha-equivalence

$$\boxed{a \# _\alpha\ u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \# _\alpha\ u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

$$\langle_a\, a\, {}_a\rangle =_\alpha \langle_b\, b\, {}_b\rangle$$

$${}_a\rangle\langle_c\ c\langle_a\ a\ {}_c\rangle\langle_b\ a\ {}_a\rangle a$$

Alpha-equivalence

$$\boxed{a \# u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \# u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

$$\langle_a\, a\, {}_a\rangle =_\alpha \langle_b\, b\, {}_b\rangle$$

$$_a\rangle\langle_c\, c\langle_a\, a\, {}_c\rangle\langle_b\, a\, {}_a\rangle a \quad =_\alpha \quad {}_a\rangle\langle_d\, d\langle_a\, a\, {}_d\rangle\langle_b\, a\, {}_a\rangle a$$

Alpha-equivalence

$$\boxed{a \# _\alpha\ u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \# _\alpha\ u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

$$\langle_a\, a\, {}_a\rangle =_\alpha \langle_b\, b\, {}_b\rangle$$

$$\begin{aligned} {}_a\rangle\langle_c\, c\langle_a\, a\, {}_c\rangle\langle_b\, a\, {}_a\rangle a &=_\alpha {}_a\rangle\langle_d\, d\langle_a\, a\, {}_d\rangle\langle_b\, a\, {}_a\rangle a \\ &=_\alpha {}_a\rangle\langle_d\, d\langle_c\, c\, {}_d\rangle\langle_b\, c\, {}_c\rangle a \end{aligned}$$

Alpha-equivalence

$$\boxed{a \#_\alpha u \triangleq \mathcal{F}_a(u) = 1.}$$

$$\frac{}{u =_\alpha u} \qquad \frac{v =_\alpha u}{u =_\alpha v} \qquad \frac{u =_\alpha v \quad v =_\alpha w}{u =_\alpha w} \qquad \frac{u =_\alpha v \quad u' =_\alpha v'}{uu' =_\alpha vv'}$$

$$a, b \#_\alpha u \quad \Rightarrow \quad u =_\alpha (a\ b) \cdot u$$

$$\langle_a\, a\, {}_a\rangle =_\alpha \langle_b\, b\, {}_b\rangle$$

$$\begin{aligned}
{}_a\rangle\langle_c\, c\langle_a\, a\, {}_c\rangle\langle_b\, a\, {}_a\rangle a &=_\alpha {}_a\rangle\langle_d\, d\langle_a\, a\, {}_d\rangle\langle_b\, a\, {}_a\rangle a \\
&=_\alpha {}_a\rangle\langle_d\, d\langle_c\, c\, {}_d\rangle\langle_b\, c\, {}_c\rangle a \\
&=_\alpha {}_a\rangle\langle_a\, a\langle_c\, c\, {}_a\rangle\langle_b\, c\, {}_c\rangle a
\end{aligned}$$

What we know

☞ Nominal transducer $\mathcal{T}$ to decide $=_\alpha$:

$$[] \xrightarrow{\langle_a / \langle_b}_{\mathcal{T}} [(a, b)] \xrightarrow{a/b}_{\mathcal{T}} [(a, b)] \xrightarrow{a\rangle / \ _b\rangle}_{\mathcal{T}} []$$

$$\rightsquigarrow \langle_a \, a \, _a\rangle =_\alpha \langle_b \, b \, _b\rangle$$

What we know

- Nominal transducer $\mathcal{T}$ to decide $=_\alpha$:

$$[] \xrightarrow{\langle_a / \langle_b}_{\mathcal{T}} [(a,b)] \xrightarrow{a/b}_{\mathcal{T}} [(a,b)] \xrightarrow{a\rangle / \ b\rangle}_{\mathcal{T}} []$$

$$\rightsquigarrow \langle_a \, a \, {}_a\rangle =_\alpha \langle_b \, b \, {}_b\rangle$$

- Memory finite regular languages:

$$[\![e]\!] \subseteq_\alpha [\![f]\!] \overset{\Delta}{\Longleftrightarrow} \forall u \in [\![e]\!], \exists v \in [\![f]\!] : \ u =_\alpha v ?$$

What we know

☞ Nominal transducer $\mathcal{T}$ to decide $=_\alpha$:

$$[] \xrightarrow{\langle_a / \langle_b}_{\mathcal{T}} [(a,b)] \xrightarrow{a/b}_{\mathcal{T}} [(a,b)] \xrightarrow{a\rangle / \ b\rangle}_{\mathcal{T}} []$$

$$\rightsquigarrow \langle_a \, a \, {}_a\rangle =_\alpha \langle_b \, b \, {}_b\rangle$$

program running in bounded memory

☞ Memory finite regular languages:

$$[\![e]\!] \subseteq_\alpha [\![f]\!] \overset{\Delta}{\Longleftrightarrow} \forall u \in [\![e]\!], \exists v \in [\![f]\!] : \ u =_\alpha v ?$$

What we know

- Nominal transducer $\mathcal{T}$ to decide $=_\alpha$:

$$[] \xrightarrow{\langle_a / \langle_b}_{\mathcal{T}} [(a,b)] \xrightarrow{a/b}_{\mathcal{T}} [(a,b)] \xrightarrow{a\rangle / \ b\rangle}_{\mathcal{T}} []$$

$$\rightsquigarrow \langle_a \, a \, {}_a\rangle =_\alpha \langle_b \, b \, {}_b\rangle$$

- Memory finite regular languages:

$$[\![e]\!] \subseteq_\alpha [\![f]\!] \overset{\Delta}{\Longleftrightarrow} \forall u \in [\![e]\!], \exists v \in [\![f]\!] : \ u =_\alpha v ?$$

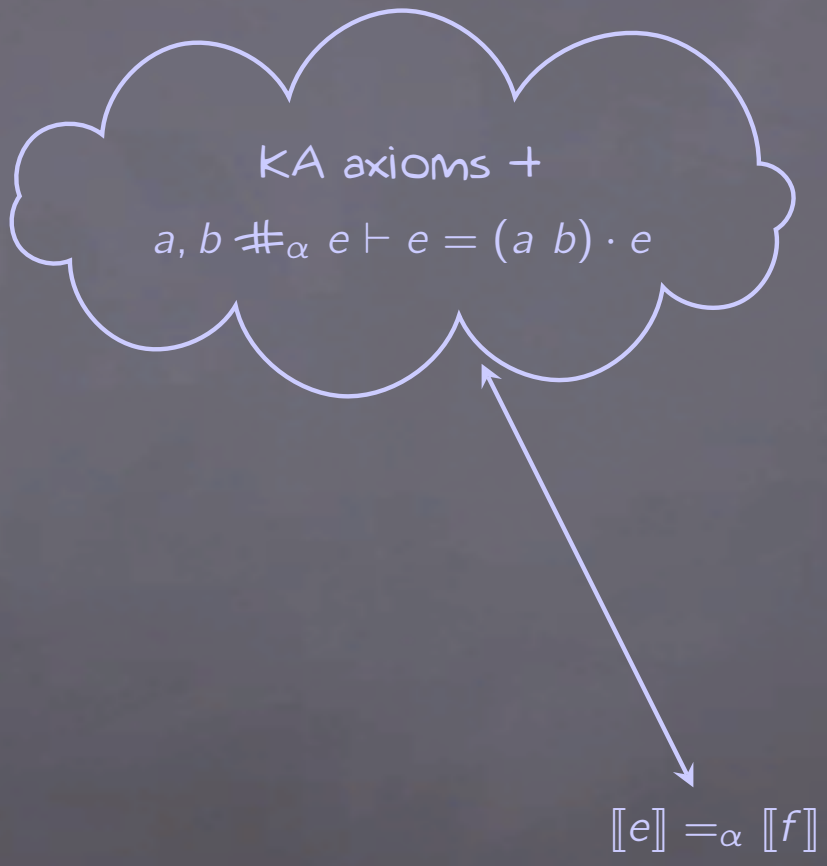
- "Normalisation" procedure, leading to a free relational interpretation.

$$(\!|u|\!) \subseteq \mathrm{Rel}\,(\Sigma^\star) \text{ such that } (\!|u|\!) = (\!|v|\!) \Leftrightarrow u =_\alpha v$$

Ongoing + Future work

Future : Ongoing

- ☞ Kleene Algebra with Binders;



Ongoing + Future work

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;

Future

Ongoing + Future work

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;
- more sophisticated software verification applications.

Future

Ongoing + Future work

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;
- more sophisticated software verification applications.

Future

- Kleene Algebra with Tests & Binders;

Ongoing + Future work

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;
- more sophisticated software verification applications.

Future

- Kleene Algebra with Tests & Binders;
- Concurrent Kleene Algebra with tests & Binders;

Ongoing + Future work

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;
- more sophisticated software verification applications.

Future

- Kleene Algebra with Tests & Binders;
- Concurrent Kleene Algebra with tests & Binders;
- even richer verification frameworks...

Ongoing + Future work

Thank you!

Ongoing

- Kleene Algebra with Binders;
- richer relational model, feat. heaps;
- more sophisticated software verification applications.

Future

- Kleene Algebra with Tests & Binders;
- Concurrent Kleene Algebra with tests & Binders;
- even richer verification frameworks...

See more at: `http://paul.brunet-zamansky.fr`