On the Computation Power of Name Parameterization in Higher-order Processes

Xian Xu[1] Qiang Yin[2] Huan Long[2]

[1]East China University of Science and Technology

[2]BASICS, Shanghai Jiao Tong Universiy

ICE 2015, Grenoble

Parameterization

It comprises abstraction and application (like those in λ-calculus).

- Π_n^D: parameterization on processes
  Higher-order abstraction: $\langle X_1, X_2, ..., X_n \rangle P$
  $(\lambda X_1, X_2, ..., X_n.P)$
  Application: $P\langle Q_1, Q_2, ..., Q_n \rangle$
  ($X_1, X_2, ..., X_n$ are process variables)

Parameterization

It comprises abstraction and application (like those in λ-calculus).

- Π_n^D: parameterization on processes
  Higher-order abstraction: $\langle X_1, X_2, ..., X_n \rangle P$
  $(\lambda X_1, X_2, ..., X_n.P)$
  Application: $P\langle Q_1, Q_2, ..., Q_n \rangle$
  ($X_1, X_2, ..., X_n$ are process variables)
- Π_n^d: parameterization on names
  First-order abstraction: $\langle x_1, x_2, ..., x_n \rangle P$
  Application: $P\langle u_1, u_2, ..., u_n \rangle$
  ($x_1, x_2, ..., x_n$ are name variables)

Parameterization

It comprises abstraction and application (like those in λ-calculus).

- Π_n^D: parameterization on processes
  Higher-order abstraction: $\langle X_1, X_2, ..., X_n \rangle P$
  $(\lambda X_1, X_2, ..., X_n.P)$
  Application: $P\langle Q_1, Q_2, ..., Q_n \rangle$
  ($X_1, X_2, ..., X_n$ are process variables)
- Π_n^d: parameterization on names
  First-order abstraction: $\langle x_1, x_2, ..., x_n \rangle P$
  Application: $P\langle u_1, u_2, ..., u_n \rangle$
  ($x_1, x_2, ..., x_n$ are name variables)

Π^d(resp. Π^D) is the union of Π_n^d(resp. Π_n^D).

Parameterization

It comprises abstraction and application (like those in λ-calculus).

- Π_n^D: parameterization on processes
  Higher-order abstraction: $\langle X_1, X_2, ..., X_n \rangle P$
  $(\lambda X_1, X_2, ..., X_n.P)$
  Application: $P\langle Q_1, Q_2, ..., Q_n \rangle$
  ($X_1, X_2, ..., X_n$ are process variables)
- Π_n^d: parameterization on names
  First-order abstraction: $\langle x_1, x_2, ..., x_n \rangle P$
  Application: $P\langle u_1, u_2, ..., u_n \rangle$
  ($x_1, x_2, ..., x_n$ are name variables)

Π^d(resp. Π^D) is the union of Π_n^d(resp. Π_n^D).

Motivation

Related Work

- Extending Π with parameterization provides strictly more expressive power. [Lanese, Pérez, Sangiorgi and Schmitt, 2010]

Motivation

Related Work

- Extending Π with parameterization provides strictly more expressive power. [Lanese, Pérez, Sangiorgi and Schmitt, 2010]
- Π extended with relabelling operator is able to encode π. [Thomsen, 1993][Xu, 2009]

Motivation

Related Work

- Extending Π with parameterization provides strictly more expressive power. [Lanese, Pérez, Sangiorgi and Schmitt, 2010]
- Π extended with relabelling operator is able to encode π. [Thomsen, 1993][Xu, 2009]
- HOcore is Turing-complete. [Lanese, Pérez, Sangiorgi and Schmitt, 2011]

Motivation

Related Work

- Extending Π with parameterization provides strictly more expressive power. [Lanese, Pérez, Sangiorgi and Schmitt, 2010]
- Π extended with relabelling operator is able to encode π. [Thomsen, 1993][Xu, 2009]
- HOcore is Turing-complete. [Lanese, Pérez, Sangiorgi and Schmitt, 2011]
- Π is not complete.[Fu, 2011]

Motivation

Related Work

- Extending Π with parameterization provides strictly more expressive power. [Lanese, Pérez, Sangiorgi and Schmitt, 2010]
- Π extended with relabelling operator is able to encode π. [Thomsen, 1993][Xu, 2009]
- HOcore is Turing-complete. [Lanese, Pérez, Sangiorgi and Schmitt, 2011]
- Π is not complete.[Fu, 2011]

Questions

- Parameterization in Π leads to completeness?
- Relative expressiveness power with π?

Outline

- Completeness of Π^d
- Relative expressiveness power of Π^d with name-passing π

Calculus Π^d

Syntax

$$E, E' := 0 \mid X \mid a(X).E \mid \overline{a}E'.E \mid E|\, E' \mid (a)E \mid \langle\widetilde{x}\rangle E \mid E\langle\widetilde{m}\rangle$$

Calculus Π^d

Syntax

$$E, E' := 0 \;\Big|\; X \;\Big|\; a(X).E \;\Big|\; \overline{a}E'.E \;\Big|\; E|\,E' \;\Big|\; (a)E \;\Big|\; \langle\widetilde{x}\rangle E \;\Big|\; E\langle\widetilde{m}\rangle$$

Operational Semantics

$$\frac{}{a(X).F \xrightarrow{a(E)} F\{E/X\}} \qquad \frac{}{\overline{a}E.F \xrightarrow{\overline{a}E} F} \qquad \frac{E \xrightarrow{\lambda} E'}{E|\,F \xrightarrow{\lambda} E'|\,F}\; bn(\lambda)\cap fn(F)=\emptyset$$

$$\frac{E \xrightarrow{a(E_1)} E' \quad F \xrightarrow{(\widetilde{c})\overline{a}[E_1]} F'}{E|\,F \xrightarrow{\tau} (\widetilde{c})(E'\mid F')} \qquad \frac{E \xrightarrow{(\widetilde{c})\overline{a}[E_1]} E'}{(d)E \xrightarrow{(d)(\widetilde{c})\overline{a}[E_1]} E'}\; d\in fn(E_1)-\{\widetilde{c},a\}$$

$$\frac{E \xrightarrow{\lambda} E'}{(c)E \xrightarrow{\lambda} (c)E'}\; c\notin n(\lambda) \qquad \frac{F \equiv E,\; E \xrightarrow{\lambda} E',\; E' \equiv F'}{F \xrightarrow{\lambda} F'}$$

The Completeness of Π^d

Calculus $\mathbb{C}$ (Fu, 2011)

$\mathbb{C}$ is a minimal model with bare primitive for computability and interactability.

Calculus $\mathbb{C}$ (Fu, 2011)

$\mathbb{C}$ is a minimal model with bare primitive for computability and interactability.

$$
\begin{array}{rcll}
P & := & 0 & \\
 & | & \Omega & \text{divergence} \\
 & | & F_a^b(f(x)) & f(x) \text{ is a computable function} \\
 & | & \overline{a}(\underline{i}) & \text{output natural number } i \\
 & | & P|P & \text{parallel composition}
\end{array}
$$

Calculus $\mathbb{C}$ (Fu, 2011)

$\mathbb{C}$ is a minimal model with bare primitive for computability and interactability.

$$
\begin{array}{rcll}
P & := & 0 & \\
 & | & \Omega & \text{divergence} \\
 & | & F_a^b(f(x)) & f(x) \text{ is a computable function} \\
 & | & \overline{a}(\underline{i}) & \text{output natural number } i \\
 & | & P|P & \text{parallel composition}
\end{array}
$$

Operational Semantics

$$\frac{}{\overline{a}(\underline{i}) \xrightarrow{\overline{a}(\underline{i})} 0} \qquad \frac{}{\Omega \xrightarrow{\tau} \Omega} \qquad \frac{P \xrightarrow{\overline{a}(\underline{i})} P' \quad Q \xrightarrow{a(\underline{i})} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \frac{P \xrightarrow{\lambda} P'}{P|Q \xrightarrow{\lambda} P'|Q}$$

$$\frac{}{F_a^b(f(x)) \xrightarrow{a(\underline{m})} \overline{b}(\underline{n})} f(\underline{m}) = \underline{n} \qquad \frac{}{F_a^b(f(x)) \xrightarrow{a(\underline{m})} \Omega} f(\underline{m}) \text{ is undefined}$$

Calculus $\mathbb{C}$ (cont'd)

- A functional $\mathbb{C}$-processes $F_a^b(f(x))$ specify nothing about how to implement it.

Calculus $\mathbb{C}$ (cont'd)

- A functional $\mathbb{C}$-processes $F_a^b(f(x))$ specify nothing about how to implement it.
- Interaction is no more than sending and receiving values.

Calculus $\mathbb{C}$ (cont'd)

- A functional $\mathbb{C}$-processes $F_a^b(f(x))$ specify nothing about how to implement it.
- Interaction is no more than sending and receiving values.
- Behavioral equality on $\mathbb{C}$ is extremely simple.

Calculus $\mathbb{C}$ (cont'd)

- A functional $\mathbb{C}$-processes $F_a^b(f(x))$ specify nothing about how to implement it.
- Interaction is no more than sending and receiving values.
- Behavioral equality on $\mathbb{C}$ is extremely simple.

Theorem (Fu, 2011)

The standard bisimulation $\simeq_{\mathbb{C}}$ coincides with $\equiv_{\mathbb{C}}$.

Calculus $\mathbb{C}$ (cont'd)

- A functional $\mathbb{C}$-processes $F_a^b(f(x))$ specify nothing about how to implement it.
- Interaction is no more than sending and receiving values.
- Behavioral equality on $\mathbb{C}$ is extremely simple.

Theorem (Fu, 2011)

The standard bisimulation $\simeq_{\mathbb{C}}$ coincides with $\equiv_{\mathbb{C}}$.

Definition

The structural congruence $\equiv_{\mathbb{C}}$ is the least congruence satisfying

$$0|\,P \equiv_{\mathbb{C}} P,\ P|\,Q \equiv_{\mathbb{C}} Q|\,P,\ (P|\,Q)|\,R \equiv_{\mathbb{C}} P|\,(Q|\,R),\ \Omega|\,\Omega \equiv_{\mathbb{C}} \Omega$$

Completeness

We say a model $\mathbb{M}$ is complete if $\mathbb{C} \sqsubseteq \mathbb{M}$.

	Turing-complete	Complete
CCS	YES	
Π	YES	
π	YES	
VPC	YES	

Completeness

We say a model $\mathbb{M}$ is complete if $\mathbb{C} \sqsubseteq \mathbb{M}$.

	Turing-complete	Complete
CCS	YES	NO
Π	YES	NO
π	YES	YES
VPC	YES	YES

Encode $\mathbb{C}$ in Π^d

- Nature numbers in Π^d are parameterizations.

$$\begin{array}{ccc} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

Encode $\mathbb{C}$ in Π^d

- Nature numbers in Π^d are parameterizations.

$$\begin{array}{lcl} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

Example. $Q \stackrel{\text{def}}{=} a(X)((i,j)(X\langle i,j\rangle \mid i.P_1 \mid j.P_2))$

Encode $\mathbb{C}$ in Π^d

- Nature numbers in Π^d are parameterizations.

$$\begin{array}{lcl} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

  Example. $Q \stackrel{\text{def}}{=} a(X)((i,j)(X\langle i,j\rangle \mid i.P_1 \mid j.P_2))$

- Encode programs instead of computable functions.

Encode $\mathbb{C}$ in Π^d

- Nature numbers in Π^d are parameterizations.

$$\begin{array}{lcl} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

  Example. $Q \stackrel{\text{def}}{=} a(X)((i,j)(X\langle i,j\rangle \mid i.P_1 \mid j.P_2))$

- Encode programs instead of computable functions.

$$\begin{array}{ccccc} \mathbb{C} & & \longrightarrow & & \Pi^d \\ \vdots & & & & \vdots \\ F_a^b(f(x)) & \longrightarrow & F_a^b(P_f) & \longrightarrow & [\![F_a^b(P_f)]\!] \end{array}$$

Encode $\mathbb{C}$ in Π^d

- Nature numbers in Π^d are parameterizations.

$$\begin{array}{lcl} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

  Example. $Q \stackrel{\text{def}}{=} a(X)((i,j)(X\langle i,j\rangle \mid i.P_1 \mid j.P_2))$

- Encode programs instead of computable functions.

$$\begin{array}{ccccc} \mathbb{C} & & \longrightarrow & & \Pi^d \\ \vdots & & & & \vdots \\ F_a^b(f(x)) & \longrightarrow & F_a^b(P_f) & \longrightarrow & [\![F_a^b(P_f)]\!] \end{array}$$

- Recursive functions as programs.
  - 3 initial functions: zero, successor and projection.
  - 3 operators: composition, minimization and recursion.

Initial Functions

$$\begin{array}{lcl} [\![\underline{0}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{y} 0 \\ [\![\underline{n+1}]\!] & \stackrel{\text{def}}{=} & \langle x, y\rangle \overline{x} [\![\underline{n}]\!] \end{array}$$

- Zero function Z_a^b

$$Z_a^b \stackrel{\text{def}}{=} a(X).\bar{b}[\![\underline{0}]\!]$$

- Successor function Suc_a^b

$$Suc_a^b \stackrel{\text{def}}{=} a(X).\bar{b}(\langle x, y\rangle \overline{x} X)$$

- Projection function $Pr(i)^b_{a_1,a_2,\ldots,a_n}$

$$Pr(i)^b_{a_1,a_2,\ldots,a_n} \stackrel{\text{def}}{=} a_1(X_1).a(X_2)\ldots a_n(X_n).\bar{b}X_i$$

Composition

$$Cmp(F, \widetilde{G})(\widetilde{x}) \stackrel{\text{def}}{=} F(G_1(\widetilde{x}), \ldots, G_k(\widetilde{x}))$$

$Cmp(F, \widetilde{G})^b_{a_1 \ldots, a_n}$ is defined by

initialize first compute each ${}^lG$ with ports $c_{l1} \ldots c_{ln}$ and b_l, then compute F and output	$(c_{11} \cdots c_{1n}, \ldots c_{k1} \ldots c_{kn}, b_1 \ldots b_k)$ $a_1(X_1).a_2(X_2). \cdots .a_n(X_n).($ $\overline{c}_{11}X_1. \cdots .\overline{c}_{1n}X_n \mid {}^1G^{b_1}_{c_{11} \ldots c_{1n}} \mid \ldots$ $\overline{c}_{k1}X_1. \cdots .\overline{c}_{kn}X_n \mid {}^kG^{b_k}_{c_{k1} \ldots c_{kn}} \mid$ $F^b_{b_1 \ldots b_k})$

Minimization

$$\mu(F)(\widetilde{x}) \stackrel{\text{def}}{=} \text{“the least } y \text{ such that } F(\widetilde{x}, y) = 0''$$

$\mu(F)^b_{a_1 \ldots a_n}$ is defined by

	$(c_1 \ldots c_n, d, e, f, g)a_1(X_1).a_2(X_2).\ldots.a_n(X_n)$
initialize	$(\overline{c}_1 X_1.\overline{c}_2 X_2.\ldots.\overline{c}_n X_n.\overline{d}[\![\underline{0}]\!] \mid !F^f_{c_1\ldots,c_n,d} \mid \overline{e}[\![\underline{0}]\!] \mid$
test the value of F	$!(i,j)f(Y)(Y\langle i,j\rangle \mid$
non-zero	$i.\overline{s}.\overline{c}_1 X_1.\overline{c}_2 X_2.\ldots.\overline{c}_n X_n \mid$
zero, output	$j.e(X).\overline{b}X)) \mid$
increase and try again	$!(s.Suc^g_e \mid g(X).\overline{d}X.\overline{e}X))$

Recursion

$$\begin{aligned} Rec(F,G)(\widetilde{x},y) &\stackrel{\text{def}}{=} H(\widetilde{x},y) \\ H(\widetilde{x},0) &= F(\widetilde{x}) \\ H(\widetilde{x},n+1) &= G(H(\widetilde{x},n),\widetilde{x},n) \end{aligned}$$

$Rec(F,G)^b_{a_1\ldots a_n,a}$ is defined by

	$(c_1\ldots c_n,d,e,f,g,h)a_1(X_1).\ldots.a_n(X_n).a(X$
initialize	$(!\overline{c}_1X_1.\ldots.\overline{c}_nX_n \mid \overline{d}X \mid F^f_{c_1\ldots c_n} \mid$
increase from zero and try	$\overline{g}[\![\underline{0}]\!] \mid !Suc^g_e \mid$
invariance ($Y+Z=n_a$)	$!(i,j,k)f(X).g(Y).d(Z).(Z\langle i,j\rangle \mid$
non-zero, call G	$i(Z').(G^f_{h,c_1\ldots c_n,k} \mid \overline{k}Y.\overline{e}Y \mid \overline{h}X \mid \overline{d}Z') \mid$
zero	$j.\overline{b}X))$

Full Abstraction

Proposition

For a k-ary recursive function $f(x_1, \ldots, x_k)$, let $F^b_{a_1,\ldots,a_k}$ be its interpretation. Then on any input $(n_1, \ldots, n_k)$, if $f(n_1, \ldots, n_k) = m$ then

$$(a_1, \ldots, a_k)(F^b_{a_1,\ldots,a_k} \mid \overline{a_1}([\![\underline{n_1}]\!]), \ldots, \mid \overline{a_k}([\![\underline{n_k}]\!])) =_{\Pi^d} \bar{b}([\![\underline{m}]\!])$$

and if $f(n_1, \ldots, n_k)$ is undefined then

$$(a_1, \ldots, a_k)(F^b_{a_1,\ldots,a_k} \mid \overline{a_1}([\![\underline{n_1}]\!]), \ldots, \mid \overline{a_k}([\![\underline{n_k}]\!])) =_{\Pi^d} !\tau$$

Full Abstraction

Proposition

For a k-ary recursive function $f(x_1, \ldots, x_k)$, let $F^b_{a_1,\ldots,a_k}$ be its interpretation. Then on any input $(n_1, \ldots, n_k)$, if $f(n_1, \ldots, n_k) = m$ then

$$(a_1, \ldots, a_k)(F^b_{a_1,\ldots,a_k} \mid \overline{a_1}(\llbracket \underline{n_1} \rrbracket), \ldots, \mid \overline{a_k}(\llbracket \underline{n_k} \rrbracket)) =_{\Pi^d} \bar{b}(\llbracket \underline{m} \rrbracket)$$

and if $f(n_1, \ldots, n_k)$ is undefined then

$$(a_1, \ldots, a_k)(F^b_{a_1,\ldots,a_k} \mid \overline{a_1}(\llbracket \underline{n_1} \rrbracket), \ldots, \mid \overline{a_k}(\llbracket \underline{n_k} \rrbracket)) =_{\Pi^d} !\tau$$

Theorem

Let P and Q be two $\mathbb{C}$-processes, it holds that

$$P \simeq_{\mathbb{C}} Q \quad \textit{if and only if} \quad \llbracket P \rrbracket =_{\Pi^d} \llbracket Q \rrbracket.$$

Discussion: Completeness of Π^D

Conjecture: NO!

- Insufficient control power of Π^D.
- Technical difficulty exists in employing the method of proving Π is not complete.

Expressiveness of Π^d

Encode Π^d in π

- Processes as names pointing to them.
- Those names act as triggers. [Sangiorgi, 1992]

$$[\![\overline{u}[\langle x_1, ..., x_n\rangle Q']. P]\!] \stackrel{def}{=} (f)(\overline{u}f.[\![P]\!] \mid !f(z).z(x_1).\cdots.z(x_n).[\![Q']\!])$$

$$[\![u(X).(X\langle d_1, ..., d_n\rangle \mid P)]\!] \stackrel{def}{=} u(x).(\overline{x}(g).\overline{g}d_1.\cdots.\overline{g}d_n \mid [\![P]\!])$$

Encode Π^d in π

- Processes as names pointing to them.
- Those names act as triggers. [Sangiorgi, 1992]

$$[\![\overline{u}[\langle x_1, ..., x_n\rangle Q']. P]\!] \quad \stackrel{def}{=} \quad (f)(\overline{u}f.[\![P]\!] \mid !f(z).z(x_1).\cdots.z(x_n).[\![Q']\!])$$

$$[\![u(X).(X\langle d_1, ..., d_n\rangle \mid P)]\!] \quad \stackrel{def}{=} \quad u(x).(\overline{x}(g).\overline{g}d_1.\cdots.\overline{g}d_n \mid [\![P]\!])$$

Theorem

Let P, Q be Π^d processes, $P =_{\Pi^d} Q$ iff $[\![P]\!] =_\pi [\![Q]\!]$.

Encode π in Π^d

- Names as processes.

- Name u is mapped to a gadget called u-pipe. [Thomsen, 1993]

$$\mathbf{u\text{-}pipe} \stackrel{def}{=} \langle x_1, x_2, x_3 \rangle (\; x_1.u(Z).\overline{x_3}Z.0 \quad \text{input}$$
$$\mid x_2.x_3(Z).\overline{u}Z.0 \quad \text{output}$$
$$)$$

Encode π in Π^d

- Names as processes.

- Name u is mapped to a gadget called u-pipe. [Thomsen, 1993]

$$\mathbf{u{-}pipe} \stackrel{def}{=} \langle x_1, x_2, x_3\rangle(\; x_1.u(Z).\overline{x_3}Z.0 \quad \text{input}$$
$$\mid x_2.x_3(Z).\overline{u}Z.0 \quad \text{output}$$
$$)$$

$$ouput:$$
$$[\![\overline{u}v.P]\!] \stackrel{def}{=} (i)(o)(c)(\mathbf{u{-}pipe}\langle i, o, c\rangle \mid \overline{o}.\overline{c}[\mathbf{v{-}pipe}].[\![P]\!])$$

$$input:$$
$$[\![u(x).P]\!] \stackrel{def}{=} (i)(o)(c)(\mathbf{u{-}pipe}\langle i, o, c\rangle \mid \overline{i}.c(X_x).[\![P]\!])$$

Completeness and Soundness

Lemma (Completeness)

Let P, Q be π processes, then $[\![P]\!] =_{\Pi^d} [\![Q]\!]$ implies $P =_\pi Q$.

Completeness and Soundness

Lemma (Completeness)

Let P, Q be π processes, then $[\![P]\!] =_{\Pi^d} [\![Q]\!]$ implies $P =_\pi Q$.

Lemma (Weak Soundness)

Let P, Q be π processes, then $P =_\pi Q$ implies $[\![P]\!] \approx_p [\![Q]\!]$.

Completeness and Soundness

Lemma (Completeness)

Let P, Q be π processes, then $[\![P]\!] =_{\Pi^d} [\![Q]\!]$ implies $P =_\pi Q$.

Lemma (Weak Soundness)

Let P, Q be π processes, then $P =_\pi Q$ implies $[\![P]\!] \approx_p [\![Q]\!]$.

$\approx_p$ differs from $=_{\Pi^d}$ in that only pipes are communicated;
we do not know if $\approx_p$ coincides with $=_{\Pi^d}$.

Conclusion

- Π^d is complete.
  - A full abstraction encoding from $\mathbb{C}$ to Π^d
  - $\mathbb{C}$ is a minimal mode with bare primitive for computability and interactability.
- Relative expressiveness of Π^d with π.
  - A full abstraction encoding from Π^d to π
  - An encoding from π to Π^d

Thanks.

June 6, 2015