Scalability using effects

Dominique Duval
with J.-C. Reynaud, J.-G. Dumas, L. Fousse, C. Domínguez

LJK, University of Grenoble

June 26, 2013
SLS 2013, Cambridge

Scalability

"*The scalability of a computer system is its ability to adapt to increased demands*"

increased = larger, more complex

Scalability

"*The scalability of a computer system is its ability to adapt to increased demands*"

increased = larger, more complex

$$\boxed{\text{small}} \rightsquigarrow \boxed{\text{LARGE}}$$

Scalability

"*The scalability of a computer system is its ability to adapt to increased demands*"

increased = larger, more complex

$$\boxed{\text{small}} \rightsquigarrow \boxed{\text{LARGE}}$$

Example.

- *The plural form of most nouns is created by adding the letter 's' to the end of the word...*
- *... but there are some exceptions...*

The "luring" trick

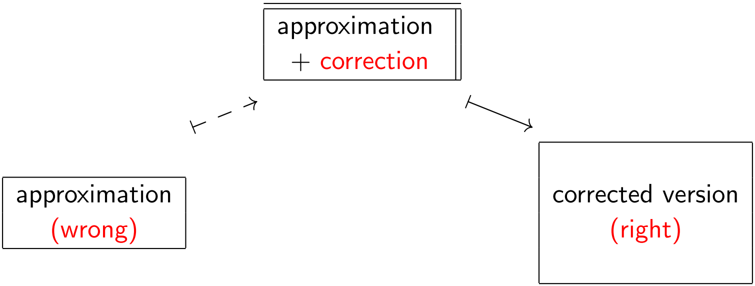
This is a well-known pedagogical trick for teaching complex features:

- first lure the students with some approximation...
- ... then add the required corrections

The "luring" trick

This is a well-known pedagogical trick for teaching complex features:

- first lure the students with some approximation...
- ... then add the required corrections



Scalability in language specification

The "luring" trick is commonly used in programming languages:
exceptions and other computational effects

Effects can be formalized in the framework of category theory:

- Monads and Lawvere theories
  for looking up and updating states, for raising exceptions,...
  [Moggi 91, Wadler 92, Haskell, Plotkin&Power 02]
- and handlers
  for handling exceptions,...
  [Plotkin&Pretnar 09]

This approach has been compared to the effect systems
[Lucassen&Gifford 88, Wadler&Thiemann 03]

In this talk

We propose a candidate for a formal language specification framework which might scale up when applied to large languages.

This framework:

- scales up thanks to a formalization of the "luring" trick
- *may* sometimes use monads or comonads, e.g.
  - the monad $TX = X + E$ for exceptions
  - the COmonad $TX = X \times S$ for states
- provides a proof system
- is based on category theory

Exceptions: operations and terms

Syntax: $f : X \to Y$

Denotation:

- f is pure if $[[f]] : [[X]] \to [[Y]]$
- f may raise exceptions if $[[f]] : [[X]] \to [[Y]] + E$
- f may catch exceptions if $[[f]] : [[X]] + E \to [[Y]] + E$

Exceptions: operations and terms

Syntax: $f\colon X \to Y$

Denotation:

- f is pure if $[[f]]\colon [[X]] \to [[Y]]$
- f may raise exceptions if $[[f]]\colon [[X]] \to [[Y]] + E$
- f may catch exceptions if $[[f]]\colon [[X]] + E \to [[Y]] + E$

Decorations (or annotations):

- $f^{(0)}$ if f is pure
- $f^{(1)}$ if f may raise exceptions ($f^{(1)}$ is called a propagator)
- $f^{(2)}$ if f may recover from exceptions ($f^{(2)}$ is called a catcher)

Conversions:

- pure $\Longrightarrow$ propagator $\Longrightarrow$ catcher

The “luring” trick for exceptions: operations

$$\begin{array}{l} f^{(0)} : X \to Y \\ f^{(1)} : X \to Y \\ f^{(2)} : X \to Y \end{array}$$

choose a decoration

$$\boxed{f : X \to Y}$$

explain the decoration

$$\begin{array}{l} f : X \to Y \\ f : X \to Y + E \\ f : X + E \to Y \end{array}$$

The "luring" trick for exceptions: composition of propagators

$$\boxed{\begin{tikzcd} X \arrow[r, "f^{(1)}"] \arrow[rr, bend right, "(g\circ f)^{(1)}"'] & Y \arrow[r, "g^{(1)}"] & Z \end{tikzcd}}$$

$$\boxed{\begin{tikzcd} X \arrow[r, "f"] \arrow[rr, bend right, "g\circ f"'] & Y \arrow[r, "g"] & Z \end{tikzcd}}$$

$$\boxed{\begin{tikzcd} & & Z+E+E \arrow[d] \\ & Y+E \arrow[ur, "g+E"] & Z+E \\ X \arrow[ur, "f"] \arrow[urr, bend right, "g\circ f"'] & Y \arrow[ur, "g"] & \end{tikzcd}}$$

Exceptions: equations

Equations: $f \equiv g \colon X \to Y$

Here in the "worst" case: f and g are catchers
i.e., $f^{(2)}, g^{(2)} \colon X \to Y$, or $f, g \colon X + E \to Y + E$

Denotation:

- the equation is strong if $f = g \colon X + E \to Y + E$
- the equation is weak if $f|_X = g|_X \colon X \to Y + E$

Exceptions: equations

Equations: $f \equiv g \colon X \to Y$

Here in the "worst" case: f and g are catchers
i.e., $f^{(2)}, g^{(2)} \colon X \to Y$, or $f, g \colon X + E \to Y + E$

Denotation:

- the equation is strong if $f = g \colon X + E \to Y + E$
- the equation is weak if $f|_X = g|_X \colon X \to Y + E$

Decorations:

- $f^{(2)} \equiv^{(st)} g^{(2)}$ if the equation is strong
- $f^{(2)} \equiv^{(wk)} g^{(2)}$ if the equation is weak

Conversions:

- $f \equiv^{(st)} g \implies f \equiv^{(wk)} g$
- if $f^{(1)}$ and $g^{(1)}$ then $f \equiv^{(st)} g \iff f \equiv^{(wk)} g$

The “luring” trick for exceptions: equations

$$\boxed{\begin{array}{l} f \equiv^{(st)} g : X \to Y \\ f \equiv^{(wk)} g : X \to Y \end{array}}$$

choose
a decoration

$$\boxed{f \equiv g : X \to Y}$$

explain
the decoration

$$\boxed{\begin{array}{l} f \equiv g : X + E \to Y + E \\ f|_X \equiv g|_X : X \to Y + E \end{array}}$$

"Core" operations and equations for exceptions

Several exception names (or types) E_i (for $i \in I$).
For each exception name E_i with parameters of type P_i,
two operations and two equations:

ordinary value (normal)		exceptional value (abrupt)
a	$\xrightarrow{\mathtt{tag}_i}$	$\boxed{a}_i$
a	$\xleftarrow{\mathtt{untag}_i}$	$\boxed{a}_i$

$$
\begin{array}{l}
\mathtt{tag}_i \colon P_i \to \mathbb{0} \\
\mathtt{untag}_i \colon \mathbb{0} \to P_i \\
\mathtt{untag}_i \circ \mathtt{tag}_i \equiv^{(wk)} \mathtt{id}_{P_i} \\
\mathtt{untag}_i \circ \mathtt{tag}_j \equiv^{(wk)} [\,]_{P_i} \circ \mathtt{tag}_j \\
\qquad \text{when } \; j \neq i
\end{array}
\quad \longmapsto \quad
\begin{array}{l}
\mathtt{tag}_i \colon P_i \to E \\
\mathtt{untag}_i \colon E \to P_i + E \\
a \mapsto \boxed{a}_i \mapsto a \in P_i \\
a \mapsto \boxed{a}_j \mapsto \boxed{a}_j \in E
\end{array}
$$

Decorated rules for exceptions

We get a decorated inference system by adding decorations — in a proper way! — to some usual inference system. E.g.:

$$(R_1)\frac{f : \mathbb{0} \to X}{f \equiv^{(wk)} [\,]_X} \quad (R_2)\frac{f_1 \equiv^{(wk)} f_2}{g \circ f_1 \equiv^{(wk)} g \circ f_2} \quad (R_3)\frac{g_1 \equiv^{(wk)} g_2}{g_1 \circ f \equiv^{(wk)} g_2 \circ f}$$

Decorated rules for exceptions

We get a decorated inference system by adding decorations — in a proper way! — to some usual inference system. E.g.:

$$(R_1)\frac{f : \mathbb{0} \to X}{f \equiv^{(wk)} [\,]_X} \quad (R_2)\frac{f_1 \equiv^{(wk)} f_2}{g \circ f_1 \equiv^{(wk)} g \circ f_2} \quad (R_3)\frac{g_1 \equiv^{(wk)} g_2}{g_1 \circ f \equiv^{(wk)} g_2 \circ f}$$

Exercice. Prove that for all $f : \mathbb{0} \to \mathbb{0}$ not containing $\mathtt{untag}_i$:

$$\mathtt{untag}_i \circ f \circ \mathtt{tag}_i \equiv^{(wk)} \mathtt{id}_{P_i}$$

Decorated rules for exceptions

We get a decorated inference system by adding decorations — in a proper way! — to some usual inference system. E.g.:

$$(R_1)\frac{f:\mathbb{0}\to X}{f\equiv^{(wk)}[\,]_X}\quad(R_2)\frac{f_1\equiv^{(wk)}f_2}{g\circ f_1\equiv^{(wk)}g\circ f_2}\quad(R_3)\frac{g_1\equiv^{(wk)}g_2}{g_1\circ f\equiv^{(wk)}g_2\circ f}$$

Exercice. Prove that for all $f:\mathbb{0}\to\mathbb{0}$ not containing $\mathtt{untag}_i$:

$$\mathtt{untag}_i\circ f\circ\mathtt{tag}_i\equiv^{(wk)}\mathtt{id}_{P_i}$$

Proof. By induction on the structure of f

- if $f^{(1)}$ then use (R_1) and the conversions, then conclude with the axiom $\mathtt{untag}_i\circ\mathtt{tag}_i\equiv^{(wk)}\mathtt{id}$
- if $\mathtt{untag}_j$ is the first catcher in f then use the axiom $\mathtt{untag}_j\circ\mathtt{tag}_i\equiv^{(wk)}[\,]\circ\mathtt{tag}_i$ and (R_2)

“Public” operations for exceptions

The core operations are wrapped

All public operations propagate exceptions

"Public" operations for exceptions

The core operations are wrapped

All public operations propagate exceptions

For raising exceptions: $\mathtt{tag}_i : P_i \to \mathbb{0}$ gives rise to $\mathtt{throw}_{i,X} : P_i \to X$ for each X:

$$P_i \xrightarrow{\mathtt{throw}_{i,X}} X \quad = \quad P_i \xrightarrow{\mathtt{tag}_i} \mathbb{0} \xrightarrow{[\,]_X} X$$

"Public" operations for exceptions

The core operations are wrapped

All public operations propagate exceptions

For raising exceptions: $\mathtt{tag}_i : P_i \to \mathbb{0}$ gives rise to $\mathtt{throw}_{i,X} : P_i \to X$ for each X:

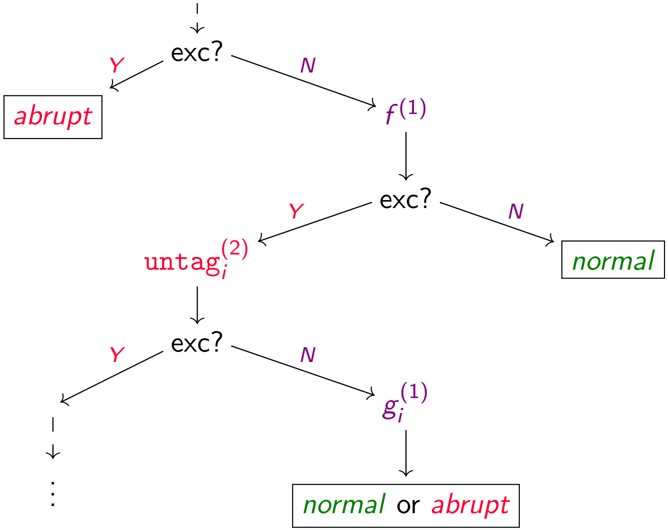
$$P_i \xrightarrow{\mathtt{throw}_{i,X}} X \quad = \quad P_i \xrightarrow{\mathtt{tag}_i} \mathbb{0} \xrightarrow{[\,]_X} X$$

For handling exceptions: $\mathtt{untag}_i : \mathbb{0} \to P_i$ gives rise to $\mathtt{try}(f)\mathtt{catch}(E_i \Rightarrow g_i | \ldots) : X \to Y$
for each $f : X \to Y$, $g_i : P_i \to Y \ldots$:

$$X \xrightarrow{\mathtt{try}(f)\mathtt{catch}(E_i \Rightarrow g_i | \ldots)} Y \quad = \quad \ldots$$

Control flow for $\texttt{try}(f)\texttt{catch}(E_i \Rightarrow g_i | \dots)$

All conditions are "`exc?`": "*is this value an exception?*"



The “luring” trick for exceptions: exc?

$$
\begin{tikzcd}
X \arrow[d, "\mathtt{id}"'] \arrow[dr, "f"] & \\
X \arrow[r, dashed] \arrow[r, phantom, shift left=3, "\equiv"] \arrow[r, phantom, shift right=3, "\equiv"'] & Y \\
\mathbb{0} \arrow[u, "{[\,]}"] \arrow[ur, "g"'] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
X \arrow[d, "\mathtt{id}"'] \arrow[dr, "f"] & \\
X \arrow[r, dashed] \arrow[r, phantom, shift left=3, "\equiv"] \arrow[r, phantom, shift right=3, "\equiv"'] & Y \\
\mathbb{0} \arrow[u, "{[\,]}"] \arrow[ur, "g"'] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
X \arrow[d] \arrow[dr, "f"] & \\
X + E \arrow[r, dashed] \arrow[r, phantom, shift left=3, "\equiv"] \arrow[r, phantom, shift right=3, "\equiv"'] & Y + E \\
E \arrow[u] \arrow[ur, "g"'] &
\end{tikzcd}
$$

Monad and Comonad for exceptions

Syntax: $f \colon X \to Y$

Denotation:

- $f^{(0)}$ is pure if $f \colon X \to Y$
- $f^{(1)}$ may raise exceptions if $f \colon X \to Y + E$
- $f^{(2)}$ may catch exceptions if $f \colon X + E \to Y + E$

Monad and Comonad for exceptions

Syntax: $f : X \to Y$

Denotation:

- $f^{(0)}$ is pure if $f : X \to Y$
- $f^{(1)}$ may raise exceptions if $f : X \to Y + E$
- $f^{(2)}$ may catch exceptions if $f : X + E \to Y + E$

With (co)monads:

- $f^{(0)} : X \to Y$ is in the base category $\mathbf{C}_0$
  $T(X) = X + E$ is a monad on $\mathbf{C}_0$
- $f^{(1)} : X \to Y$ is in the Kleisli category $\mathbf{C}_1 = \mathbf{Kl}(\mathbf{C}_O, T)$
  $T(X) = X + E$ is a comonad on $\mathbf{C}_1$
- $f^{(2)} : X \to Y$ is in the coKleisli category $\mathbf{C}_2 = \mathbf{coKl}(\mathbf{C}_1, T)$

Comonad and Monad for states

Syntax: $f : X \to Y$

Denotation:

- $f^{(0)}$ is pure if $f : X \to Y$
- $f^{(1)}$ may observe the state if $f : X \times S \to Y$
- $f^{(2)}$ may modify the state if $f : X \times S \to Y \times S$

Comonad and Monad for states

Syntax: $f : X \to Y$

Denotation:

- $f^{(0)}$ is pure if $f : X \to Y$
- $f^{(1)}$ may observe the state if $f : X \times S \to Y$
- $f^{(2)}$ may modify the state if $f : X \times S \to Y \times S$

With (co)monads:

- $f^{(0)} : X \to Y$ is in the base category $\mathbf{C}_0$
  $T(X) = X \times S$ is a comonad on $\mathbf{C}_0$
- $f^{(1)} : X \to Y$ is in the coKleisli category $\mathbf{C}_1 = \mathbf{coKl}(\mathbf{C}_O, T)$
  $T(X) = X \times S$ is a monad on $\mathbf{C}_1$
- $f^{(2)} : X \to Y$ is in the Kleisli category $\mathbf{C}_2 = \mathbf{Kl}(\mathbf{C}_1, T)$

The “luring” trick for states: pairs

$$\begin{tikzcd} & Y_1 \\ X \arrow[ur, "f"] \arrow[r, dashed] \arrow[dr, "g"'] & Y_1 \times Y_2 \arrow[u, "\mathtt{pr}_1"'] \arrow[d, "\mathtt{pr}_2"] \arrow[u, phantom, "\equiv", near start, shift right=6] \arrow[d, phantom, "\equiv", near start, shift left=6] \\ & Y_2 \end{tikzcd}$$

$$\begin{tikzcd} & Y_1 \\ X \arrow[ur, "f"] \arrow[r, dashed] \arrow[dr, "g"'] & Y_1 \times Y_2 \arrow[u, "\mathtt{pr}_1"'] \arrow[d, "\mathtt{pr}_2"] \arrow[u, phantom, "\equiv", near start, shift right=6] \arrow[d, phantom, "\equiv", near start, shift left=6] \\ & Y_2 \end{tikzcd}$$

$$\begin{tikzcd} & Y_1 \\ X \times S \arrow[ur, "f"] \arrow[r, dashed] \arrow[dr, "g"'] & Y_1 \times Y_2 \times S \arrow[u] \arrow[d] \arrow[u, phantom, "\equiv", near start, shift right=6] \arrow[d, phantom, "\equiv", near start, shift left=6] \\ & Y_2 \times S \end{tikzcd}$$

Application: Sequential product (seq)

"*first* f_1 *then* f_2"

$$
\begin{tikzcd}
X_1 \arrow[r, "f_1"] \arrow[dr, phantom, "\equiv"] & Y_1 \arrow[r, "\mathtt{id}"] \arrow[dr, phantom, "\equiv"] & Y_1 \\
X_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \arrow[u] \arrow[d] \\
X_2 \arrow[r, "\mathtt{id}"] & X_2 \arrow[r, "f_2"] & Y_2
\end{tikzcd}
$$

Application: Sequential product (seq)

"*first* f_1 *then* f_2"

$$
\begin{tikzcd}
X_1 \arrow[r, "f_1"] \arrow[dr, phantom, "\equiv"] & Y_1 \arrow[r, "\mathtt{id}"] \arrow[dr, phantom, "\equiv"] & Y_1 \\
X_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \arrow[u] \arrow[d] \\
X_2 \arrow[r, "\mathtt{id}"] & X_2 \arrow[r, "f_2"] & Y_2
\end{tikzcd}
$$

$$
\begin{tikzcd}
X_1 \arrow[r, "f_1"] \arrow[dr, phantom, "\equiv"] & Y_1 \arrow[r, "\mathtt{id}"] \arrow[dr, phantom, "\equiv"] & Y_1 \\
X_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \arrow[u] \arrow[d] \\
X_2 \arrow[r, "\mathtt{id}"] & X_2 \arrow[r, "f_2"] & Y_2
\end{tikzcd}
\iff
\begin{tikzcd}
X_1 \arrow[r, "f_1"] \arrow[dr, phantom, "\equiv"] & Y_1 \\
X_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \arrow[u] \arrow[d] \\
X_2 \arrow[r, "f_2"] & Y_2
\end{tikzcd}
$$

Application: Sequential product (seq)

"*first* f_1 *then* f_2"

$$
\begin{tikzcd}
X_1 \arrow[r, "f_1"] & Y_1 \arrow[r, "\mathtt{id}"] & Y_1 \\
X_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[ur, phantom, "\equiv"] \arrow[dr, phantom, "\equiv"] & Y_1 \times X_2 \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[ur, phantom, "\equiv"] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \arrow[u] \arrow[d] \\
X_2 \arrow[r, "\mathtt{id}"] & X_2 \arrow[r, "f_2"] & Y_2
\end{tikzcd}
$$

$$\longmapsto$$

$$
\begin{tikzcd}
X_1 \times S \arrow[r, "f_1"] & Y_1 \times S & Y_1 \arrow[r, "\mathtt{id}"] & Y_1 \\
X_1 \times X_2 \times S \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[ur, phantom, "\equiv"] \arrow[dr, phantom, "\equiv"] & Y_1 \times X_2 \times S \arrow[u] \arrow[d] \arrow[r, dashed, "\mathtt{id}"] & Y_1 \times X_2 \times S \arrow[u] \arrow[d] \arrow[r, dashed] \arrow[ur, phantom, "\equiv"] \arrow[dr, phantom, "\equiv"] & Y_1 \times Y_2 \times S \arrow[u] \arrow[d] \\
X_2 \arrow[r, "\mathtt{id}"] & X_2 & X_2 \times S \arrow[r, "f_2"] & Y_2 \times S
\end{tikzcd}
$$

Conclusion

An effect "is" a *span*:

$$\begin{tikzcd} & \boxed{\mathit{DECORATED}} \arrow[dl, shift right] \arrow[dr] & \\ \boxed{\mathit{small}} \arrow[ur, dashed, shift right] & & \boxed{\mathrm{LARGE}} \end{tikzcd}$$

- right branch: semantics
- left branch: proofs [Coq]

(This framework relies on categorical tools:
adjunction, categories of fractions, limit sketches)

- IMPORTANT: provides a stepwise scalability method:
  the combinaison of effects "is" the composition of spans
- HOPEFULLY: compatible with other scalability methods

THANK YOU

Cited papers

Lucassen&Gifford 88 J. M. Lucassen, D. K. Gifford. Polymorphic effect systems. POPL 1988.ACM Press, p. 47-57 (1988).

Moggi 91 Eugenio Moggi. Notions of Computation and Monads. Information and Computation 93(1), p. 55-92 (1991).

Plotkin&Power 02 G. D. Plotkin, J. Power. Notions of Computation Determine Monads. FoSSaCS 2002. LNCS 2303, p. 342-356 (2002).

Plotkin&Pretnar 09 G. D. Plotkin, M. Pretnar. Handlers of Algebraic Effects. ESOP 2009. LNCS 5502, p. 80-94 (2009).

Wadler 92 P. Wadler. The essence of functional programming. POPL 1992. ACM Press, p. 1-14 (1992).

Wadler&Thiemann 03 P. Wadler, P.Thiemann. The Marriage of Effects and Monads. ACM Trans. on Computational Logic, 4, p. 1-32 (2003).

Our papers

- About exceptions and states:
  - J.-G.Dumas, D. Duval, L. Fousse, J.-C. Reynaud. A duality between exceptions and states. MSCS 22 p.719-722 (2012)
  - J.-G.Dumas, D. Duval, L. Fousse, J.-C. Reynaud. Decorated proofs for computational effects: States. ACCAT 2012. EPTCS 93 p.45-59 (2012)
  - J.-G.Dumas, D. Duval, L. Fousse, J.-C. Reynaud. Adjunctions for exceptions. arXiv:1207.1255 (2012)
- About the categorical framework:
  - C. Domínguez, D. Duval. Diagrammatic logic applied to a parameterization process. MSCS 20 p. 639-654 (2010)
  - J.-G.Dumas, D. Duval, J.-C. Reynaud. Cartesian effect categories are Freyd-categories. JSC 46 p. 272-293 (2011)