

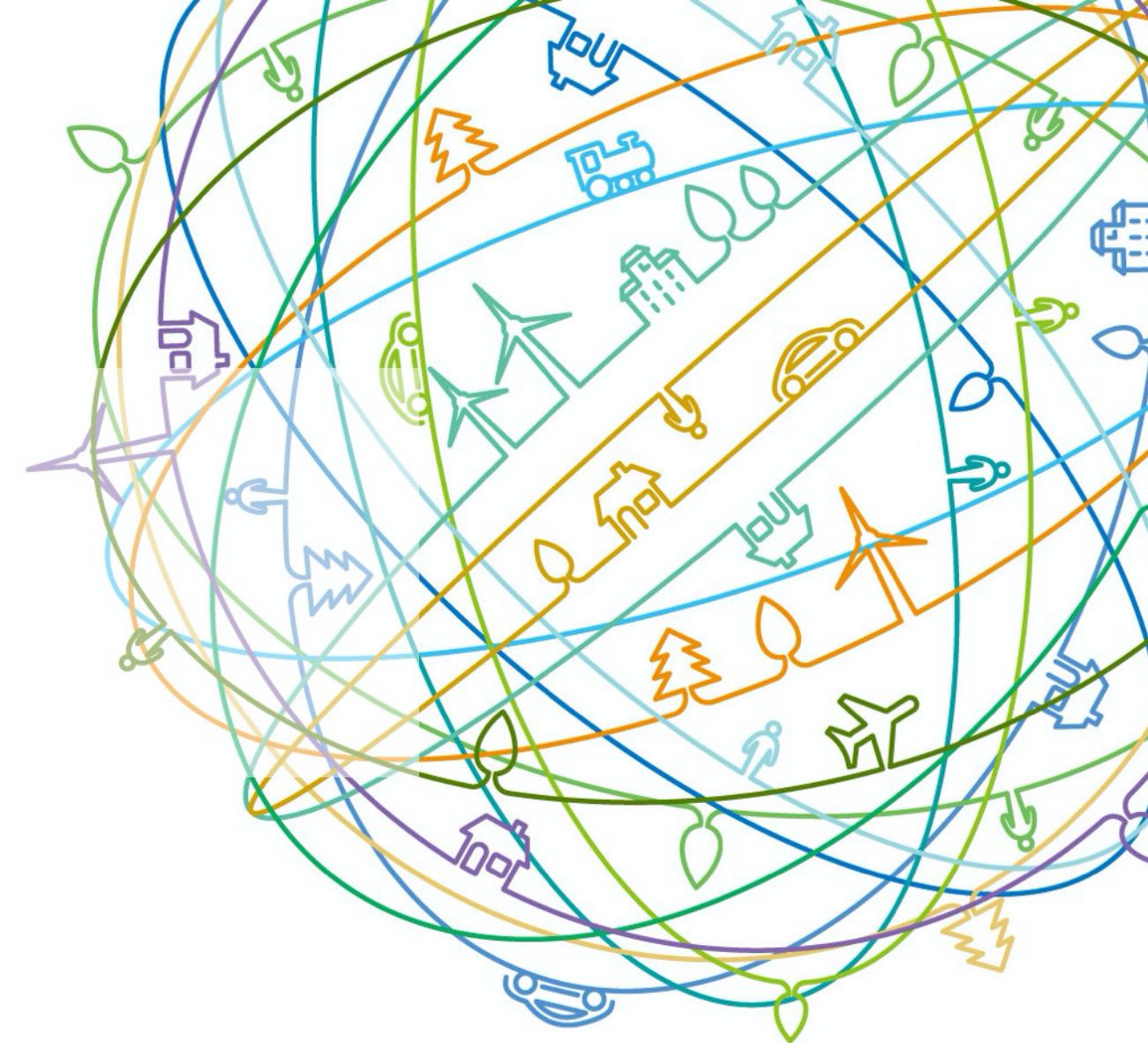
[M]<sup>s</sup>

# 自动微分 实现方式



ZOMI 酱

Building a better connected world



[www.mindspore.cn](http://www.mindspore.cn)

# 关于本课程

## 1. 课程背景

- AI框架中自动微分的重要性

## 2. 课程内容

- 微分基本概念：数值微分 - 符号微分 - 自动微分
- 自动微分模式：前向微分 – 后向微分 – 雅克比原理
- 具体实现方式：表达式或图 – 操作符重载OO – 源码转换 AST
- MindSpore实现：基于图表示的源码转换Graph Base AST
- 自动微分的未来
- 自动微分的挑战

# AD Implication Method

[M]<sup>s</sup>

## 基本表达式法 LIB

- 封装基本的表达式及其微分表达式作为库函数
- 运行时记录基本表达式和相应的组合关系
- 链式法则对基本表达式的微分结果进行组合



## 操作符重载法 OO

- 利用语言多态特性，使用操作符重载基本运算表达式
- 运行时记录基本表达式和相应的组合关系
- 链式法则对基本表达式的微分结果进行组合



## 源码转换法 AST

- 语言预处理器、编译器或解释器的扩展
- 对程序表达进行分析得到基本表达式的组合关系
- 链式法则对基本表达式的微分结果进行组合

[M]<sup>s</sup>  
MindSpore

# AD Implication Method

[M]<sup>s</sup>

## 《Automatic Differentiation in Machine Learning: a Survey》

BAYDIN, PEARLMUTTER, RADUL, AND SISKIND

24

Language	Tool	Type	Mode	Institution / Project	Reference	URL
AMPL	AMPL	INT	F, R	Bell Laboratories	Fourer et al. (2002)	<a href="http://www.ampl.com/">http://www.ampl.com/</a>
C, C++	ADIC	ST	F, R	Argonne National Laboratory	Bischof et al. (1997)	<a href="http://www.mcs.anl.gov/research/projects/adic/">http://www.mcs.anl.gov/research/projects/adic/</a>
	ADOL-C	OO	F, R	Computational Infrastructure for Operations Research	Walther and Griewank (2012)	<a href="https://projects.coin-or.org/ADOL-C">https://projects.coin-or.org/ADOL-C</a>
C++	Ceres Solver	LIB	F	Google		<a href="http://ceres-solver.org/">http://ceres-solver.org/</a>
	CppAD	OO	F, R	Computational Infrastructure for Operations Research	Bell and Burke (2008)	<a href="http://www.coin-or.org/CppAD/">http://www.coin-or.org/CppAD/</a>
	FABDAD++	OO	F, R	Technical University of Denmark	Bendtsen and Stauning (1996)	<a href="http://www.fabbad.com/fabbad.html">http://www.fabbad.com/fabbad.html</a>
	Mxyzptlk	OO	F	Fermi National Accelerator Laboratory	Ostiguy and Michelotti (2007)	
C#	AutoDiff	LIB	R	George Mason Univ., Dept. of Computer Science	Shtof et al. (2013)	<a href="http://autodiff.codeplex.com/">http://autodiff.codeplex.com/</a>
F#, C#	<b>DiffSharp</b>	OO	F, R	Maynooth University, Microsoft Research Cambridge	Baydin et al. (2016a)	<a href="http://diffsharp.github.io">http://diffsharp.github.io</a>
Fortran	ADIFOR	ST	F, R	Argonne National Laboratory	Bischof et al. (1996)	<a href="http://www.mcs.anl.gov/research/projects/adifor/">http://www.mcs.anl.gov/research/projects/adifor/</a>
	NAGWare	COM	F, R	Numerical Algorithms Group	Naumann and Riehme (2005)	<a href="http://www.nag.co.uk/nagware/Research/ad_overview.asp">http://www.nag.co.uk/nagware/Research/ad_overview.asp</a>
	TAMC	ST	R	Max Planck Institute for Meteorology	Giering and Kaminski (1998)	<a href="http://autodiff.com/tamc/">http://autodiff.com/tamc/</a>
Fortran, C	COSY	INT	F	Michigan State Univ., Biomedical and Physical Sci.	Berz et al. (1996)	<a href="http://www.bt.pa.msu.edu/index_cosy.htm">http://www.bt.pa.msu.edu/index_cosy.htm</a>
	Tapenade	ST	F, R	INRIA Sophia-Antipolis	Hascoët and Pascual (2013)	<a href="http://www-sop.inria.fr/tropics/tapenade.html">http://www-sop.inria.fr/tropics/tapenade.html</a>
Haskell	ad	OO	F, R	Haskell package		<a href="http://hackage.haskell.org/package/ad">http://hackage.haskell.org/package/ad</a>
Java	ADiJaC	ST	F, R	University Politehnica of Bucharest	Slusanschi and Dumitrel (2016)	<a href="http://adijac.cs.pub.ro">http://adijac.cs.pub.ro</a>
	Deriva	LIB	R	Java & Clojure library		<a href="https://github.com/lambder/Deriva">https://github.com/lambder/Deriva</a>
Julia	JuliaDiff	OO	F, R	Julia packages	Revels et al. (2016a)	<a href="http://www.juliadiff.org/">http://www.juliadiff.org/</a>
Lua	<b>torch-autograd</b>	OO	R	Twitter Cortex		<a href="https://github.com/twitter/torch-autograd">https://github.com/twitter/torch-autograd</a>
MATLAB	ADiMat	ST	F, R	Technical University of Darmstadt, Scientific Comp.	Willkomm and Vehreschild (2013)	<a href="http://adimat.sc.informatik.tu-darmstadt.de/">http://adimat.sc.informatik.tu-darmstadt.de/</a>
	INTLab	OO	F	Hamburg Univ. of Technology, Inst. for Reliable Comp.	Rump (1999)	<a href="http://www.ti3.tu-harburg.de/rump/intlab/">http://www.ti3.tu-harburg.de/rump/intlab/</a>
	TOMLAB/MAD	OO	F	Cranfield University & Tomlab Optimization Inc.	Forth (2006)	<a href="http://tomlab.biz/products/mad">http://tomlab.biz/products/mad</a>
Python	ad	OO	R	Python package		<a href="https://pypi.python.org/pypi/ad">https://pypi.python.org/pypi/ad</a>
	<b>autograd</b>	OO	F, R	Harvard Intelligent Probabilistic Systems Group	Maclaurin (2016)	<a href="https://github.com/HIPS/autograd">https://github.com/HIPS/autograd</a>
	<b>Chainer</b>	OO	R	Preferred Networks	Tokui et al. (2015)	<a href="https://chainer.org/">https://chainer.org/</a>
	<b>PyTorch</b>	OO	R	PyTorch core team	Paszke et al. (2017)	<a href="http://pytorch.org/">http://pytorch.org/</a>
	<b>Tangent</b>	ST	F, R	Google Brain	van Merriënboer et al. (2017)	<a href="https://github.com/google/tangent">https://github.com/google/tangent</a>
Scheme	R6RS-AD	OO	F, R	Purdue Univ., School of Electrical and Computer Eng.		<a href="https://github.com/qobi/R6RS-AD">https://github.com/qobi/R6RS-AD</a>
	Scmutils	OO	F	MIT Computer Science and Artificial Intelligence Lab.	Sussman and Wisdom (2001)	<a href="http://groups.csail.mit.edu/mac/users/gjs/6946/refman.txt">http://groups.csail.mit.edu/mac/users/gjs/6946/refman.txt</a>
	Stalingrad	COM	F, R	Purdue Univ., School of Electrical and Computer Eng.	Pearlmutter and Siskind (2008)	<a href="http://www.bcl.hamilton.ie/~qobi/stalingrad/">http://www.bcl.hamilton.ie/~qobi/stalingrad/</a>

F: Forward, R: Reverse; COM: Compiler, INT: Interpreter, LIB: Library, OO: Operator overloading, ST: Source transformation

实现以下表达式计算：

$$f(x, y, z) = (x + y)/z$$

手动将表达式函数分解为库函数中基本表达式组合：

$$a = x + y \quad b = x/y$$

库函数中定义对应表达式的数学微分规则和链式法则：

$$da = dx + dy \quad db = dx/y - dy * x/y^2$$

- 优点：**
- 实现简单
  - 任意编程语言

- 缺点：**
- 使用库函数进行编程
  - 无法使用原语言运算表达式

```
def ADAdd(x, dx, y, dy, z, dz):  
    z = x + y  
    dz = dx + dy
```



$$\begin{aligned} a &= x + y \\ da &= dx + dy \end{aligned}$$

```
def ADDiv(x, dx, y, dy, z, dz):  
    z = x / y  
    dz = dx / y + (x / (y * y)) * dy
```



$$\begin{aligned} b &= x/y \\ db &= dx/y - dy * x/y^2 \end{aligned}$$

```
>>> call ADAdd(x, dx, y, dy, a, da)  
>>> call ADDiv(x, dx, y, dy, b, db)
```

# OO , Operator Overload

[M]<sup>s</sup>

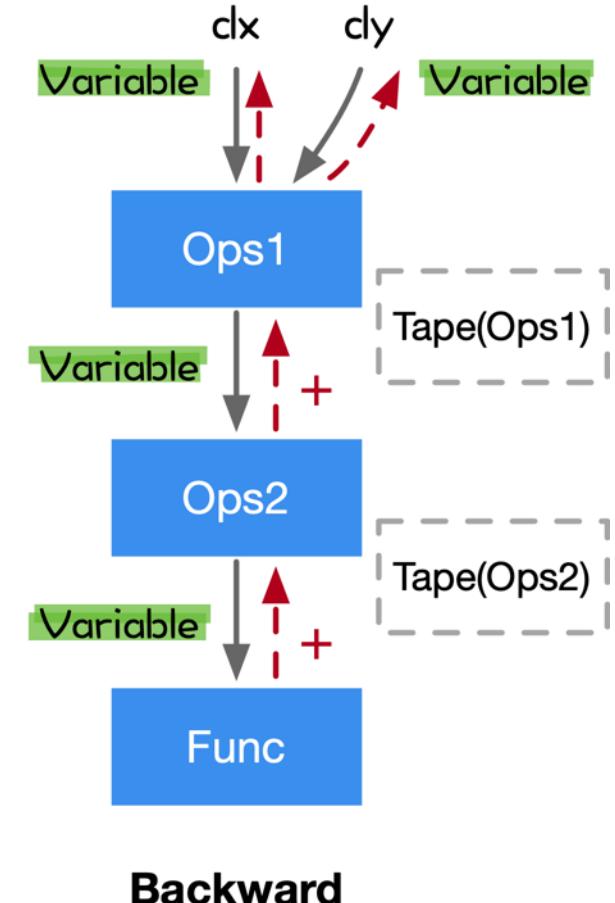
1. 利用语言的多态性，重载基本运算操作符
2. 将表达式操作类型和输入输出信息，记录到 Tape 中
3. 对 Tape 遍历，并对其中记录的基本运算操作进行微分
4. 把结果通过链式法则进行组合

**优点：**

- 实现简单
- 语言具备多态性
- 易用性高，贴合原生语言

**缺点：**

- 显式的构造 Tape 数据结构和对 Tape 进行读写
- 额外数据结构和操作的引入，不利于高阶微分
- if , while 等控制流表达式，通常难以通过操作符重载



# OO , Operator Overload

[M]<sup>s</sup>

①

```
class Variable:  
    def __mul__(self, other):  
        return ops_mul(self, other)  
    def __add__(self, other):  
        return ops_add(self, other)
```

②

```
def grad(l, results):  
    for entry in reversed(gradient_tape):  
        dl_doutputs = entry.outputs  
        dl_dinputs = entry.propagate(dl_doutputs)
```

③

```
class Tape:  
    inputs: List[str]  
    outputs: List[str]  
    propagate: d_inputs, d_outputs
```

④

```
for input, dl_din in zip(entry.inputs, dl_dinputs):  
    dl_d[input] += dl_din
```

1. 分析获得语言程序的 AST 表达形式 ( Parse )
2. 基于 AST 完成基本表达式的分解和微分操作 ( Infer )
3. 遍历 AST 得到基本表达式间的依赖关系
4. 应用链式法则组成完成自动微分

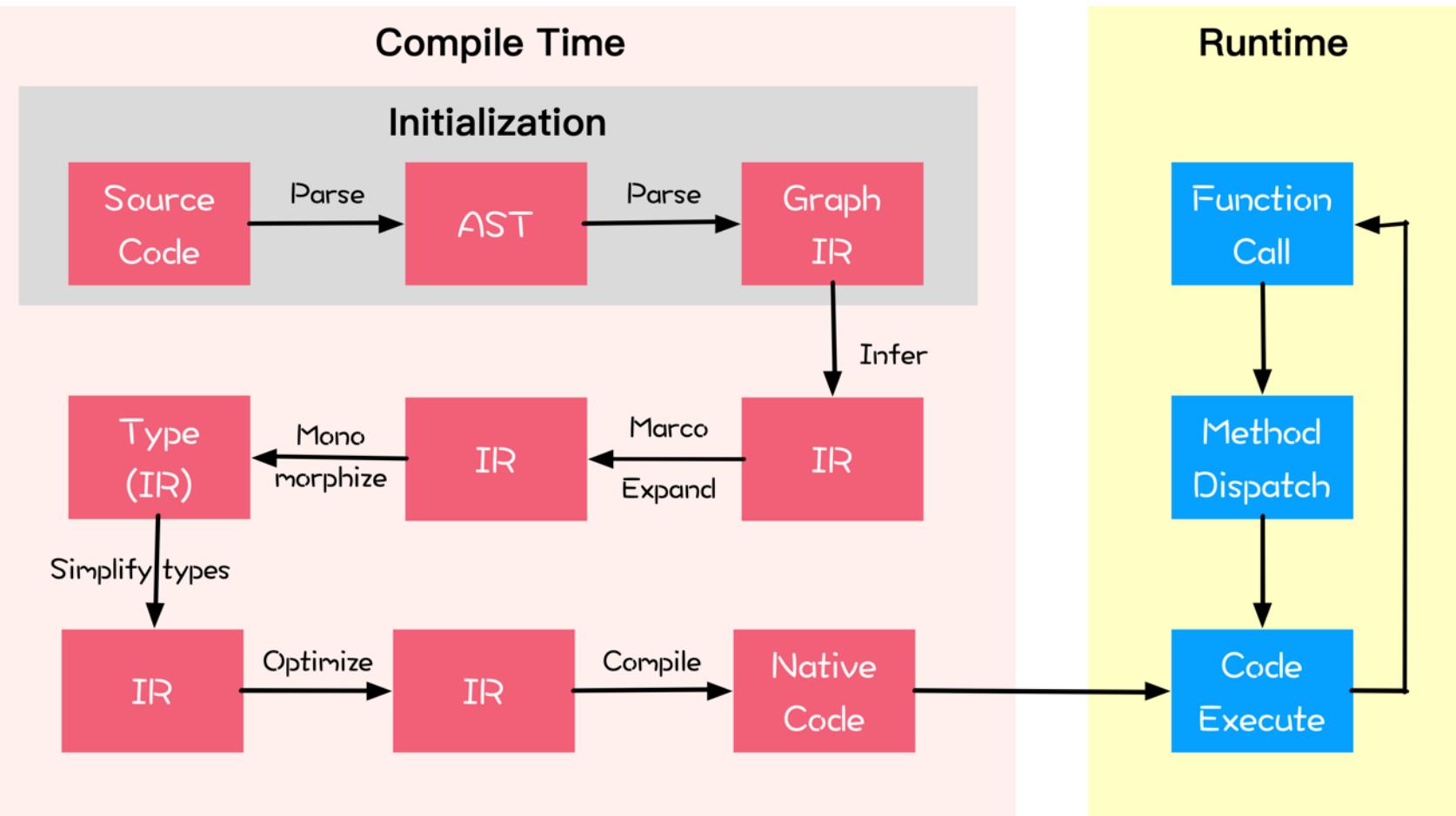
**优点：** • 丰富数据类型和与语言原生操作

- 无额外Tape等数据结构，易于实现高阶微分
- 微分结果以代码的形式存在，方便分布式系统计算

**缺点：** • 代码理解难，涉及更多计算机底层编译等原理

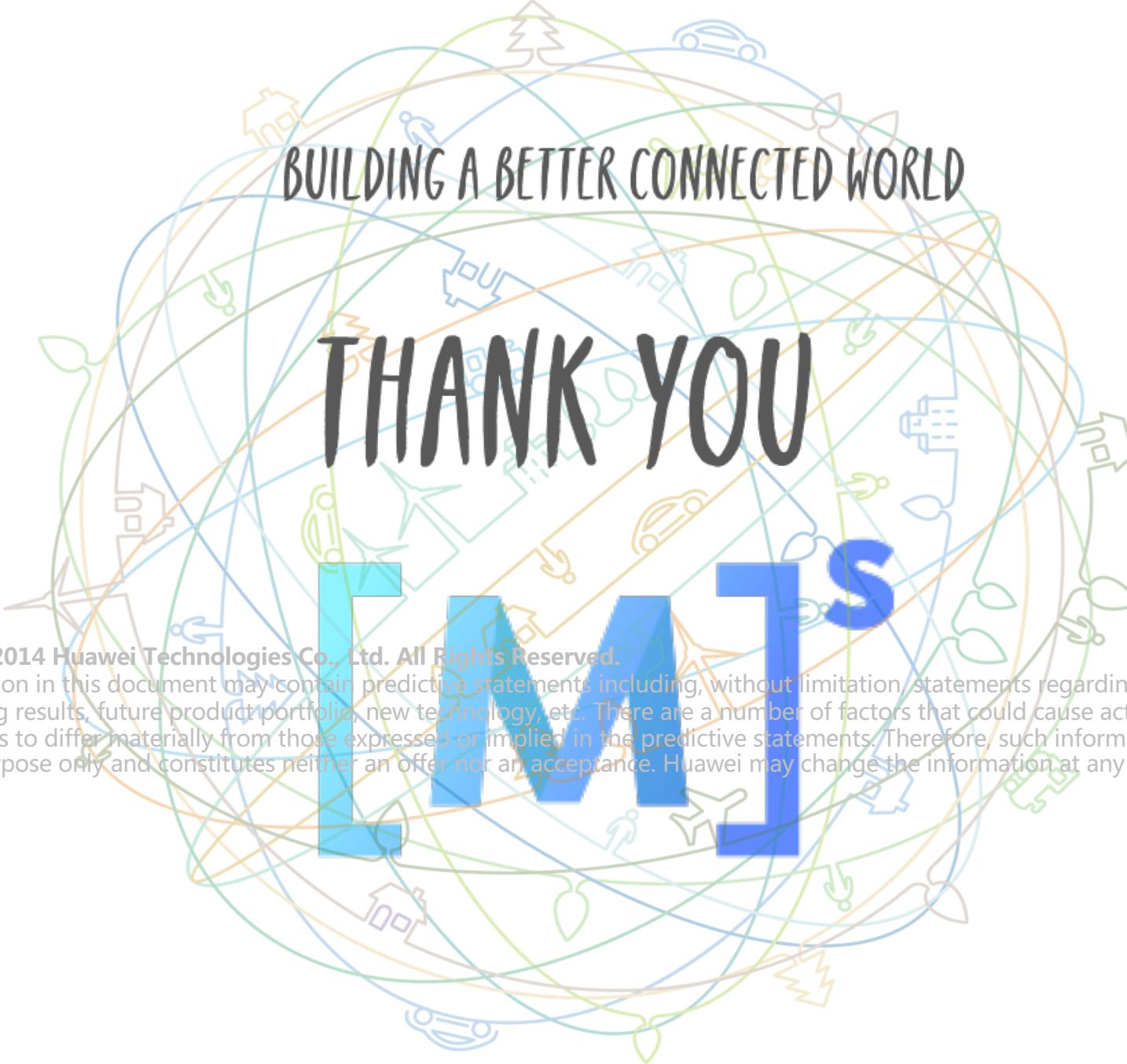
- 实现复杂度高，需要扩展语言的预处理器、编译器或解释器
- 容易写出不支持的代码导致错误，需要更强的检查告警系统

- Parse
- Infer
- Monomorphism
- Optimize
- Compile
- Execute
- Function Call



# Conclusion

1. 自动微分的实现方式可以分为三种：表达式或图、操作符重载和源码转换
2. 基于表达式实现主要依赖构建基础微分表达库，手动调用库
3. 基于操作符重载依赖于语言的多态性来记录实现
4. 基于源码转换核心在于AST完成基本表达式的分解和微分操作



BUILDING A BETTER CONNECTED WORLD

THANK YOU

TEAM

Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.