Reo

1、Reo是什么

2、Reo用来解决什么问题

3、Reo的基本语义

4、Reo的Connector之间的性质

5、基于Connector实现的简单例子

6、与工业界结合例子-产品

7、编码工具

1、Reo是什么

Reo 是一种基于通道的外生协调语言，其中复杂的协调器（称为连接器）由更简单的协调器组合而成。Reo中最简单的连接器是一组由用户提供的具有良好定义的行为的通道。Reo 中的每个连接器都对连接到它的实体（例如，组件）施加了特定的协调模式。因此，连接器本质上是对组件之间的协议或“粘合代码”进行建模。Reo 中的重点仅在于连接器及其组合，而不是通过这些连接器连接、通信和协作的实体。Reo中的每个连接器对通过该连接器执行I/O操作的实体(如组件)施加了特定的协调模式，而这些实体并不知道这些实体。

2、Reo用来解决什么

Reo是一种用于编程和分析协调协议的领域特定语言，这些协议将单个进程组合成完整的系统。也就是说Reo用来做实体之间的连接，例如应用于分布式、移动网络架构、通信拓扑在运行时的动态变化来进行组合系统，多用于面向服务的系统、多线程系统的协调。

它是用来解决多个组件间、线程间、系统内部、系统与系统间的协调（数据交互I/O），从而将各部分组合形成一个完成系统。

3、Reo的基本语义

基本概念：

组件实例：是一组非空的活动实体，可以是顺序代码的片段或模块、被动或主动对象、线程、进程、代理或软件组件，与其他实体之间的交互仅可使用I/O。

组件：组件是一种软件实现，其实例可以在物理或逻辑设备上执行。因此，组件是描述其实例属性的抽象类型。

Channel：Reo中的原子连接器，具有唯一的标识，自身无方向，但具有两端，其中source端接受数据进入channel，sink端消耗数据离开channel。若channel的一端被任何组件实例所知道，那么该组件实例中的任何实体都可以使用该channel。（仅用于传输数据，使用input/output在两端）

Connector：连接器，指的是channel或者channel的复合连接

Node：通道的末端逻辑连接点称为节点，包含三种（source节点、sink节点、混合节点）

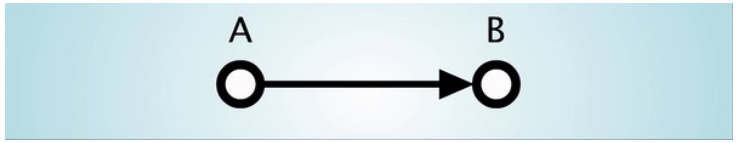
3、Reo的基本语义

每个channel自身具有一个type(完全由用户自定义)，包含create、forget、disconnect、wait和connect、move、read、take、write。这些都是channel自身内部的一些操作（每一个channel都具有这些操作）其中前半部分是无需条件，后半部分是需要条件的，基本操作如下表所示。

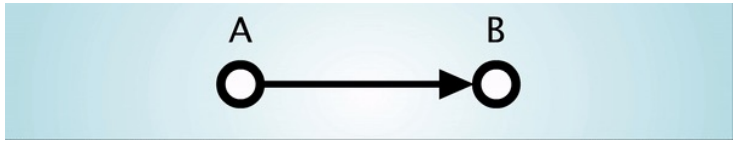
Operation	Con.	Description
_create(*chantype*)	-	Creates a channel of type *chantype* and returns the identifiers of its two channel ends.
_forget(*e*)	N	changes *e* such that it no longer refers to the channel end it designates.
_move(*e, loc*)	Y	moves *e* to the location *loc*.
_connect(*[t,] e*)	N	Connects the specified channel end, *e*, to the component instance that contains the active entity that performs this operation.
_disconnect(*e*)	N	Disconnects the specified channel end from the component instance that contains the active entity that performs this operation.
_wait(*[t,] conds*)	N	Suspends the active entity that performs this operation, waiting for the conditions specified in *conds* to become true for the specified channel ends.
_read(*[t,] inp[, v[, pat]]*)	Y	Suspends the active entity that performs this operation, waiting for a value that can match with *pat*, to become available for reading from the sink channel end *inp* into the variable *v*. The _read operation is non-destructive: the value is copied from the channel into the variable, but the original remains intact.
_take(*[t,] inp[, v[, pat]]*)	Y	This is the destructive variant of _read: the channel loses the value that is read.
_write(*[t,] outp, v*)	Y	Suspends the active entity that performs this operation, until it succeeds to write the value of the variable *v* to the source channel end *outp*.

4、 connector之间的性质

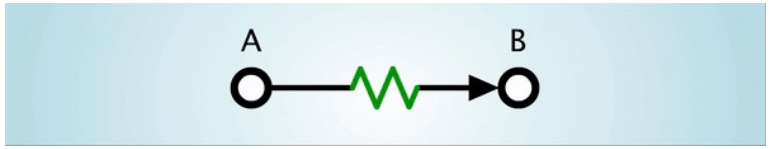
Channel的类型，例如同步通道、过滤通道、(empty/full)FIFO1(FIFOn)通道、 syncdrain、asyncdrain等。



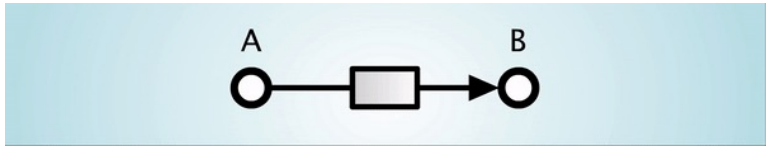
Read：如果想读取数据，那么必须等待source端有数据写入，所以B端右边读取操作一直阻塞，直到A端右端有数据写入。Write同理。如果Reader和Writer同时存在，就不会发生阻塞。

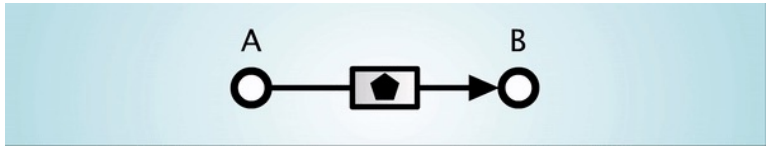


过滤通道数据未成功通过过滤，那数据传输就是失败的，其余和同步通道基本一致。

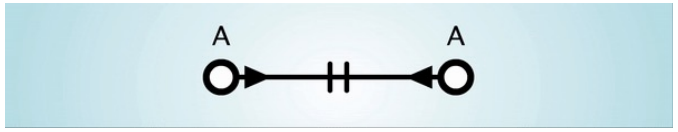


Empty/full的write操作，当缓冲器中是否有数据，其中1或者n表可缓存的个数

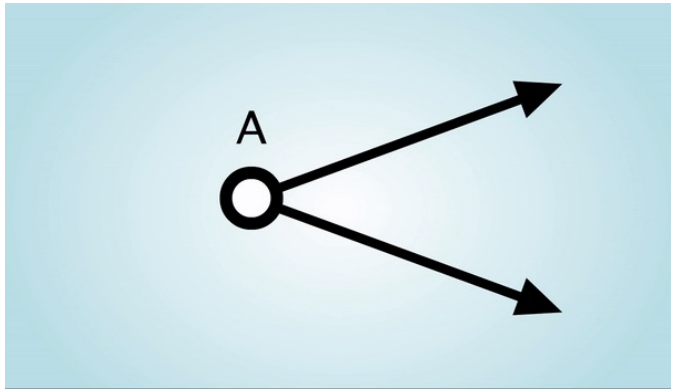


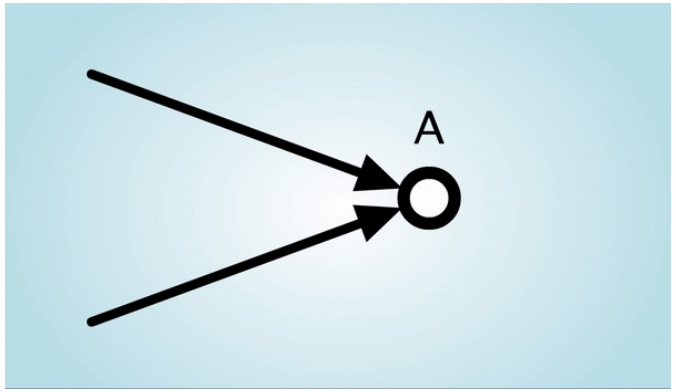


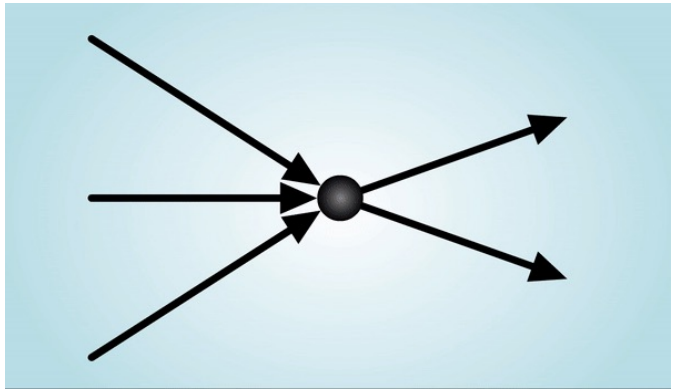
同步和异步的drain操作，就两个source端接受数据并丢弃掉，其中同步需要两端同时匹配，异步则直接丢弃。



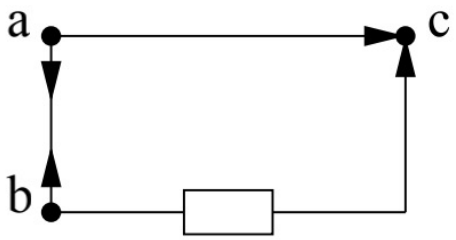
4、 connector之间的性质

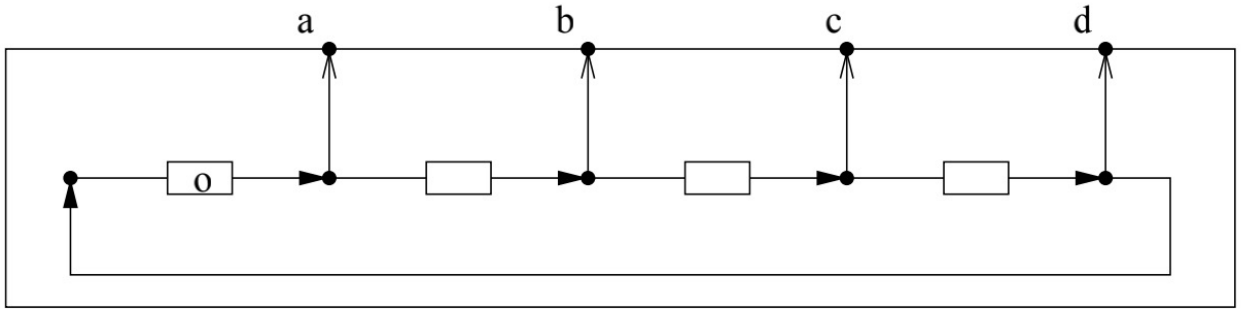


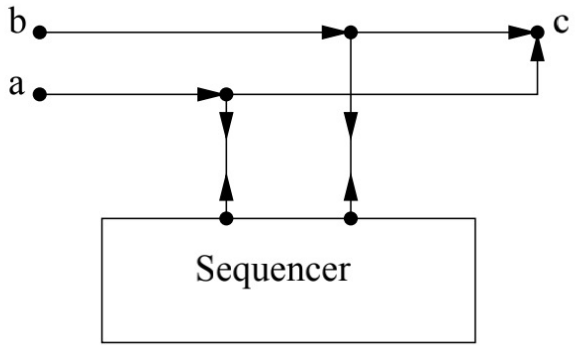


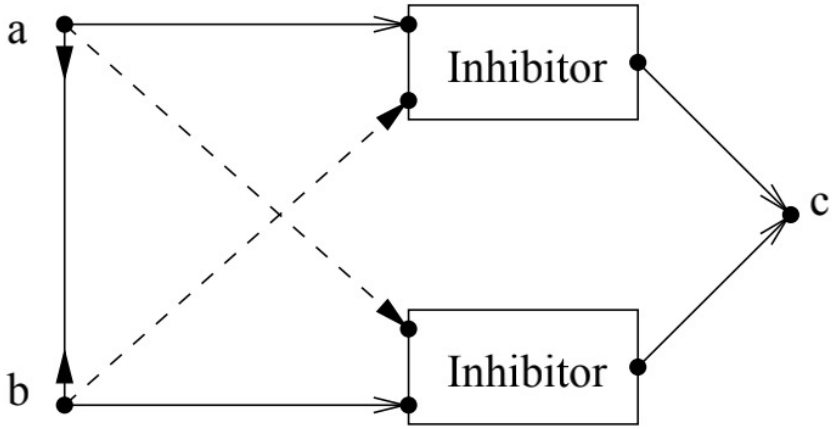


5、基于connector实现的简单例子

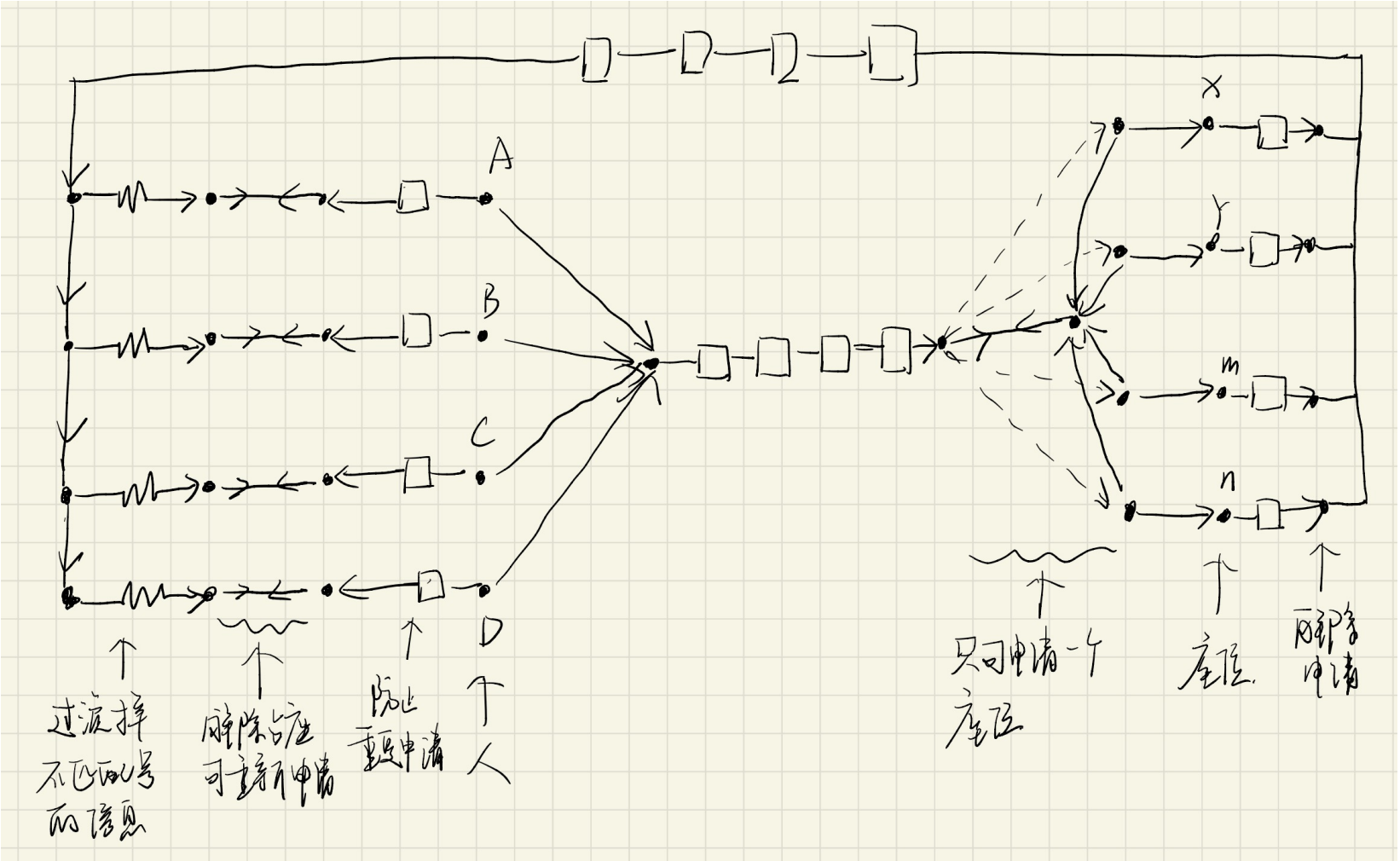








5、基于connector实现的简单例子



6、工业界结合的例子

基于 Reo 的智能城市协调协议工程解决方案

Besharati M R, Izadi M. A Reo Based Solution for Engineering the Coordination Protocols for Smart Cities[J]. arXiv preprint arXiv:2012.14280, 2020.

一种基于 Reo 的软件定义网络 (SDN) 的正式模型，其中由基本 Reo 原语组成的声明性结构组成，以指定 SDN 的数据和控制平面的描述性模型

Feng, Hui, Farhad Arbab, and Marcello Bonsangue. "A Reo Model of Software Defined Networks." In International Conference on Formal Engineering Methods, pp. 69-85. Springer, Cham, 2019.

用于自动生成 Reo 连接器的部分分布式、部分集中式实现

Jongmans S S T Q, Santini F, Arbab F. Partially distributed coordination with Reo and constraint automata[J]. Service Oriented Computing and Applications, 2015, 9(3): 311-339.

6、工业界结合的例子

应用金融领域的一个场景:控制电子银行系统的业务流程

Ter Beek M H, Gadducci F, Santini F. Validating reconfigurations of Reo circuits in an e-banking scenario[C]//Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems. 2013: 39-48.

对于每个 Web 服务，我们会自动生成一个代理来管理该服务与 Reo电路之间的通信

Jongmans S S T Q, Santini F, Sargolzaei M, et al. Automatic code generation for the orchestration of web services with Reo[C]//European Conference on Service-Oriented and Cloud Computing. Springer, Berlin, Heidelberg, 2012: 1-16.

ReoService在分布式业务流程中互连和协调服务

Koehler C, Lazovik A, Arbab F. ReoService: coordination modeling tool[C]//International Conference on Service-Oriented Computing. Springer, Berlin, Heidelberg, 2007: 625-626.

协调的业务流程由一组 Web 服务组成，这些服务的集体行为由 Reo 协调

Lazovik A, Arbab F. Using Reo for service coordination[C]//International Conference on Service-Oriented Computing. Springer, Berlin, Heidelberg, 2007: 398-403.

7、编码工具

我们最新的 Reo 编译器将一些 Reo 连接器的文本规范映射到所选目标语言的实现。当前版本的编译器支持 Java、Promela（SPIN 模型检查器的输入语言）和Maude 的代码生成。编译器可以轻松扩展以支持其他目标语言。

Java平台插件：https://github.com/ReoLanguage/Reo

Eclipse的平台扩展的插件：ECT支持： Reo连接器和（变体）约束自动机的图形编辑、使用代数图变换对连接器进行动态重新配置等，但由于打算消除对Eclipse和Flash的依赖，已经被弃用。

7、编码工具

ReoLive：旨在编译一组基于 Java/Scala 的 Reo 相关工具，使用 Web 前端与它们交互。ReoLive 是轻量级的，因为只需要一个支持 JavaScript 的离线 Internet 浏览器就可以使用它。对于更复杂的操作，还支持客户端-服务器架构。Github网站：https://github.com/ReoLanguage/ReoLive

Dreams：Dreams（“原子多个步骤的分布式运行时评估”的首字母缩写词），也称为分布式 Reo 引擎，是一个分散的、基于参与者的框架，其中协调原语在分布式环境中执行。但是该框架不再维护。

8、总结

Reo的官网自2019年7月起就没维护，其中部分工具都下线了，尤其是eclipse的插件，在其官网中的相关项目的网站地址均不正确，目前对Reo的研究仍然存在，但是在谷歌学术上21年有关Reo的文章并不多(好像不足5篇)，Reo论文集中在05-18年之间，主要应用在Web服务组合的编排协议，对于信息物理系统应用并不广泛，在大多是结合约束自动机、结合概率，对组件间的连接器进行建模、还有对系统之间进行协调（例如银行、并发游戏调度等）。