Open Source introduction and legal aspects for developers

by Carlo Piana, Array

CERN, Geneva, 27 March 2023

Presenter

Carlo Piana

IT Lawyer, Milano, Array

Digital liberities advocate

20+ years experience in FOSS

Former GC FSFE

Contributing Member, Eclipse Foundation

Member of the board, OSI

Book →

Carlo Piana

OPEN SOURCE, SOFTWARE LIBERO E ALTRE LIBERTÀ

UN'INTRODUZIONE ALLE LIBERTÀ DIGITALI

PREFAZIONE DI ROBERTO DI COSMO

POSTFAZIONE DI SIMONE ALIPRANDI

copyleft-italia.it

Ledizioni

What we’ll learn today

Take away points

If nothing else, you should take home some basic concepts:

- Free Software is a complex phenomenon, its fundamental tool is a license relying on copyright.
- The holder of copyright (not necessarily the author) has the exclusive right to authorize certain actions
- Free Software uses copyright to *permit* the 4 liberties of software, and sometimes to make sure that this permission remains untouched in *derivatives*
- Derivative software is a set of code that reuses existing code to make a larger combination or modification of the original ("*upstream*") code
- Copyright owners of original code must authorize derivatives. Authorization is given with the license. A FS license contains the authorization (grant). The grant is conditional.

Take away point (continues)

- Free Software is also commercial software. The opposite of Free (open) is proprietary
- There are many different licenses (too many!), with different conditions
- Notable licenses are MIT/BSD, Apache, {GPL, AGPL, LGPL} and MPL.
- Making software Free is an act of public display of will: different ways, but better be standard (REUSE)
- Compliance is a legal, technical concept; non compliance has consequences, but also a matter of respect of others work.
- Compliance is a process, better it be a continuous process. We introduce the concept of CI/CD/CC
- You will learn some of the tools we use

Copyright, in a nutshell

Copyright, an exclusive right

Just like all “intellectual property”, copyright is the right to exclude others from doing certain things.

- negative right: it’s a right to prohibit
- only on one specific subject matter
- one thing, multiple rights: “mine” has a peculiar meaning

Worldwide

- Copyright is obtained once for all in all the countries subject to the Berne Convention
- According to these principles:
    - No formal application
    - No special wording necessary (© BY)
    - creation of an original work
- Software is protected by copyright as a literary work (!)
- But! Always bear in mind: software has an utilitarian value

“Creation”

- Copyright only protects creative, original objects: an original form of expression
- There must be some degree of freedom:
  - No copyright on facts, data, numbers, abstract concepts
  - Interfaces are in general not copyrightable
  - The more external constraints, the less likely is copyright infringement

Main difference with patents

- copyright protects the original form of expression
- patents protect the general idea of how to resolve a technical problem
- think of a recipe and the cake
    - patent protects the cake
    - copyright protects how the recipe is written, the actual words you use
- to violate copyright (on software) I must have copied
- to violate patents, I don’t even need to know that a patent exists

Who is the owner?

- general rule: it is assigned to the author(s)
- copyright is a tradable object (in part)
  - exception: moral rights (not available everywhere)
- copyright can be assigned (=transferred onto somebody else)
- copyright can be sold, leased, pledged, rented…
- exception: work of an employee → employer
- exception: work for hire → client
  - both heavily dependant on jurisdiction

Joint ownership

When two or more authors have contributed to the work and their contribution can’t be easily separated from the others’.

But they must have done their work together

In software is a very difficult thing to ascertain: if you can undo the commits of one author?

Public domain

- the copyright holder has totally waived their copyright
- when nobody has a right to control the subject matter because:
- with the elapsing of the duration of the right (life + 70 years!)
- moral rights never elapse, can’t be waived.
- is public domain a form of free software?

Licensing

Basic concepts

- In order to copy, execute, distribute, translate, modify software you need a permission form the owner.
- This permission is contained in a legal deed called “license”
- License contains the terms under which this permission is granted
- There are contractual licenses and “bare licenses”, containing only conditions:
  - If condition is complied with, then you can {modify, distribute original or modified software}
  - If condition is not complied with, then you cannot {modify, distribute original or modified software}

Derivatives

The most important concept today

Nobody writes software from scratch!

- If software contains substantial fragments of other software, it is a derivative
- This includes libraries that are linked.
- Statically or dynamically changes little
  - statically: it is certainly a derivative
  - dynamically: it can be derivative (false myth it is not)
  - One of the most controversial issue in Compliance
  - heavily programming-language dependant too (eg C vs. Java)

This is still not FOSS

- These concepts work for any software licensing
- From a copyright point of view, no difference between not complying with a FOSS condition and not paying the price for the additional copy you make or use the higher version of a license
- Including criminal charges – there are, in some states, Italy is one of them: art. 171-bis LDA (copyright code)

Copyleft

A license that requires derivatives to be under the same or compatible license.

- All derivatives: strong copyleft
- Only the single file/library: weak (lesser) copyleft

Where are we left with all this?

- no software comes without legal tags, the legal tags being “all rights reserved”
- in order to use and reuse software one must ascertain whether they have permission to do so
- permission is granted by a license
- a license can be private (individually granted, site wide, bundled with hardware, whatever) or public
- a public license is a license given to everybody, typically it is a bare license, whether it’s a contract is a matter (damn too complicate) of applicable law
  - if you think licensing is complicate, don’t ever mess up with international private law

This is what you do:

- You take some code
- That code has its own conditions
- Conditions of software you are reusing is called inbound
- Conditions of software you are *distributing* is called outbound
- Derivative software needs permission from the original(s) owner

Compliance

“Compliance” means:

- Means *respect* conditions upon which you receive a Free Software *grant*
- But legally means that all conditions must be met or exceeded
- The more the conditions, the harder complying (no copyleft easier → strong copyleft harder)
- Compliance is a matter of distribution: there is no issue for internal use (so called ASP loophole).

Common pitfalls

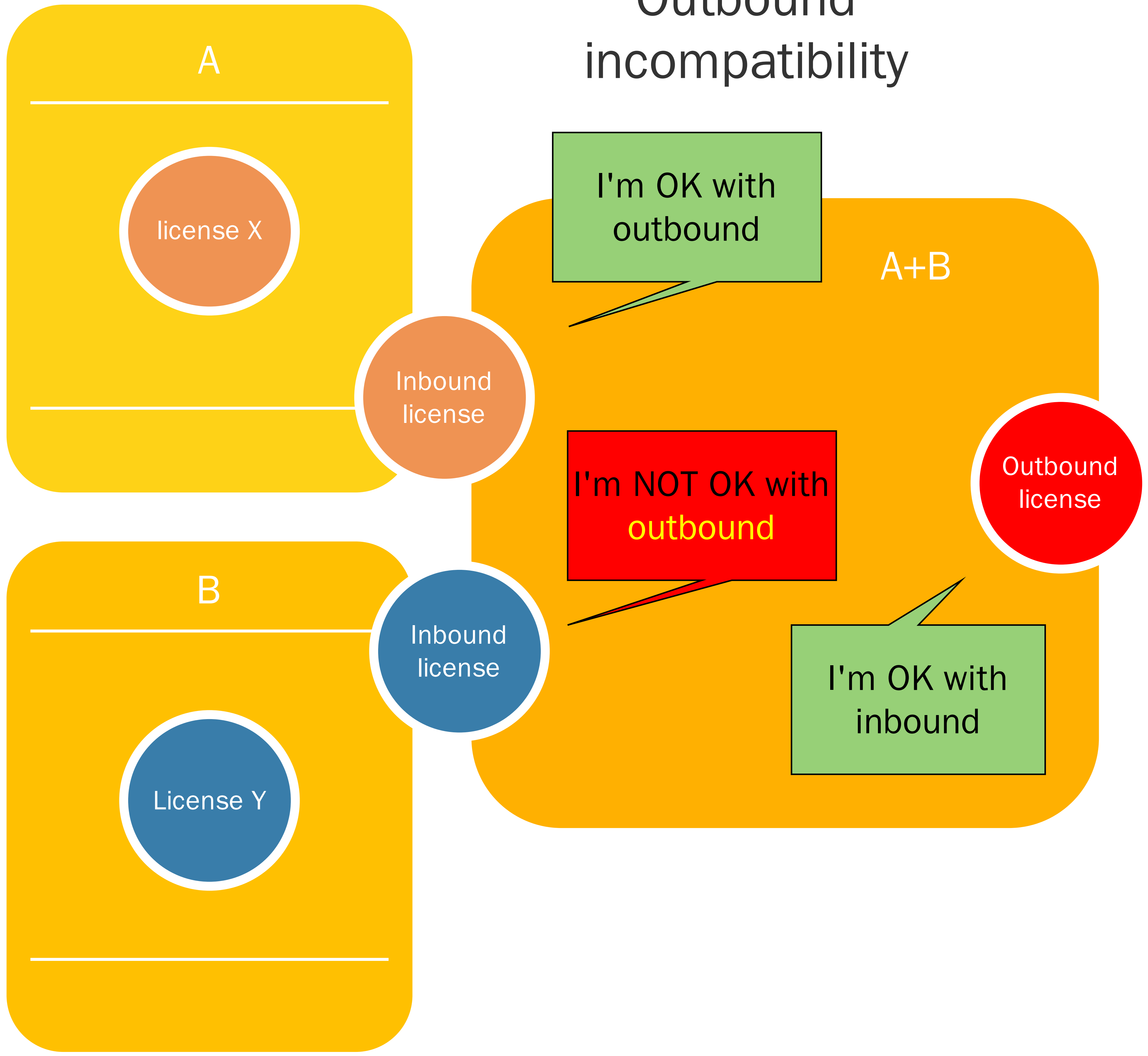
- Entire source code (copyleft)
- Attribution in the required form (all)
- Provide a Notice (all)
- Accompany the work with the license text (all)
- Combining incompatible licenses (copyleft)

A Clash of Licenses

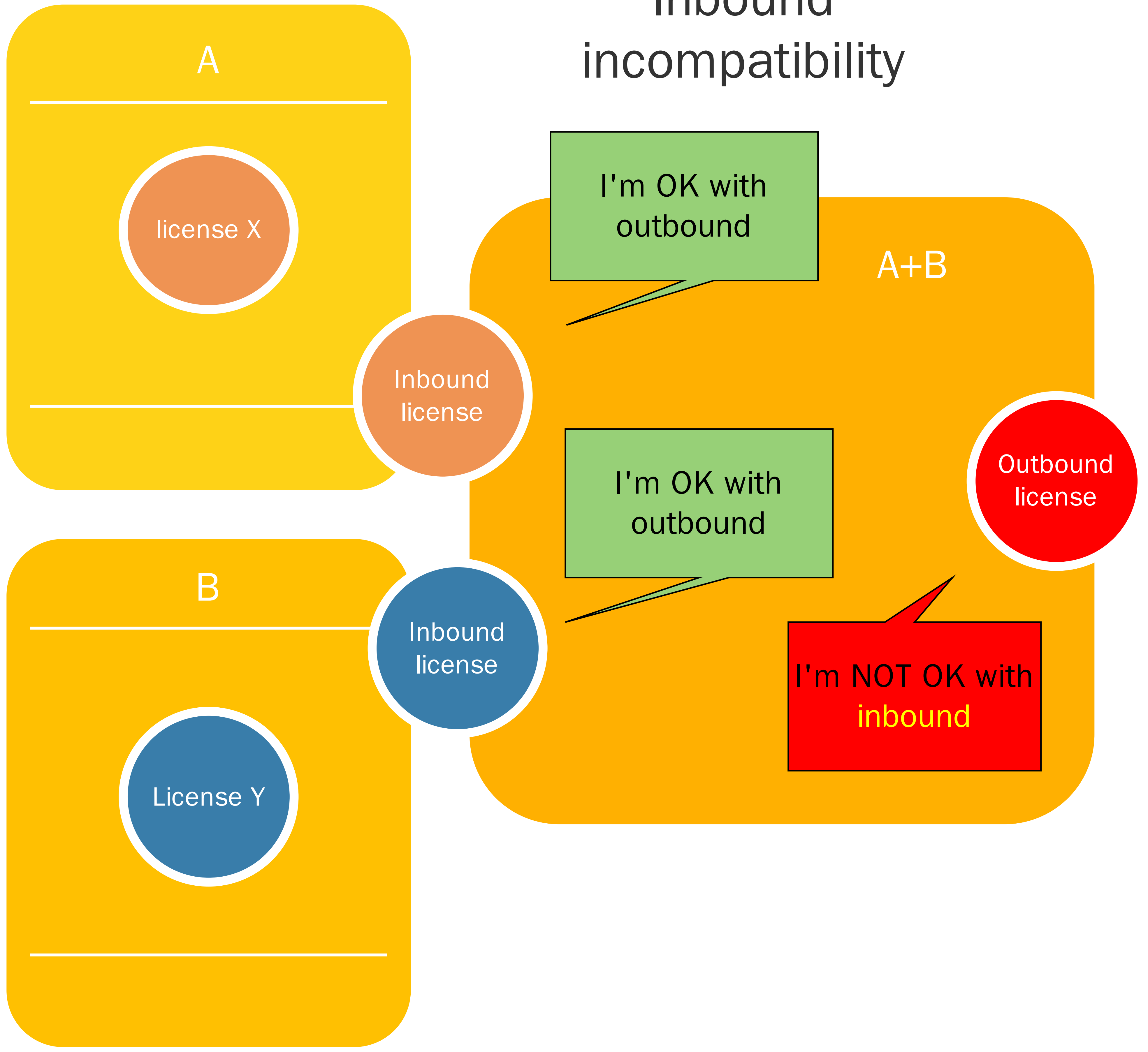
The more conditions and the stricter, the more likely you have incompatibilities: there is no state in which you can comply with both.

(in)Compatibility is no one-way road

Outbound incompatibility



Inbound incompatibility



Information on incompatible licenses?

- The more reliable is the FSF's license compatibility list (https://www.gnu.org/licenses/license-compatibility.html)
- That's only for GNU GPL

Fact: there are very few compliance cases

Why? Nobody knows.

- litigation costs
- many don’t care
- the preferred route is not litigation
- then *there are* cases, especially in Germany, USA and recently in Italy (we brought it)

How you can make sure you are in compliance?

- software selection
- license analysis
- map dependencies
- tools
- process
- CI/CD/CC

Software selection

You rely on information you have (mainly)

Sometimes this is not reliable

There are ways to make sure it’s *more* reliable

Scanning

- Code scanning for cheats (reuse of code from other projects without disclosing, such as BlackDuck)
- Code scanning to find licensing information (like Fossology, Scancode, ScanOSS [also for plagiarism])
- Code scanning can be applied upfront, at the end, but it should be part of the continuous process

There is no shortcut

- Scanning is a tool, a tool is worth if used for its purposes and within its limits
- It is a complement to a general system
- In a complex, business environment, this is not close to be enough
- a “holistic approach”

OpenChain

- A standard ISO/IEC 5230
- Standards reuse standards
  - Open chain reuses standards like SPDX (ISO/IEC 5962:2021)
- It provides for compliance artifacts, proofs that you are compliant, and have a system to ensure that you comply
- Requires process, clear definition of roles, adequate education
- Not one off, revised on a rolling basis, as any quality system
- Main purpose: giving proof that you are compliant and your partners can rely on your efforts in a chain of software provision
- Self assessment or even third-party certified
- BoM

https://www.openchainproject.org/

SPDX

- ISO/IEC 5962:2021
- A standard to describe the license applicable to software components
- in a machine readable way
- describing dependencies
- BoM

Some do's and dont's

- Do select upstream code from reliable sources
- Don't assume that because software has no license attached you can use it (reverse is true)
- Don't use upstream code without running it past tech and compliance scrutiny
- Do write your software using the official license for the project
- Don't reinvent the wheel by using a different license for your own pet projects
- Don't assume that a practice is safe because many do the same
- Do always follow guidelines, this is part of the job of a developer

- Do use REUSE as a way to apply license to your code (scans will ensure you do anyway)
- Do insist that any project you contribute to uses SPDX and REUSE to provide licensing info
- Don’t take shortcuts. You might get away with it in proprietary, but if the source code is in public display…
- Don’t just rely on scanning or on tools. They are great, but they don’t make for a wise use of software
- Do in case of doubt, refer to compliance to seek clearance
- Do it in case you want to reuse code whose licenses are not “approved”

Questions?

Thank you!