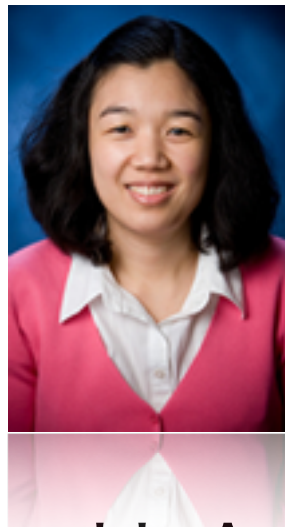


Intellectual Property Rights Requirements for Heterogeneously- Licensed Systems



Thomas A. Alspaugh
Department of Computer
Science
Georgetown University



Hazeline U. Asuncion
Institute for Software
Research University of
California



Walt Scacchi
Institute for Software
Research University of
California

背景

- ✧ 生産性, 信頼性向上のため, OSSを再利用したシステムが増加してきている
- ✧ 複数のOSSを再利用した場合, 1つのシステムに異なるライセンスが共存しているシステム
(Heterogeneously-Licensed Systems : HtLS)になる
- ✧ HtLSを合法的にリリースするには全てのOSSライセンスの制約を満たさなければならない

HtLSのライセンス遵守方法

- ✧ 著者に直接許可をもらう
 - こういうシステムを開発してるんだけど、あなたのソフトウェア使っていい？
- ✧ 含まれている全てのライセンスの義務を果たし、その権利を得る
 - 権利(right)：ライセンスが明示的に許可していること
 - 義務(obligation)：権利を侵害するもの

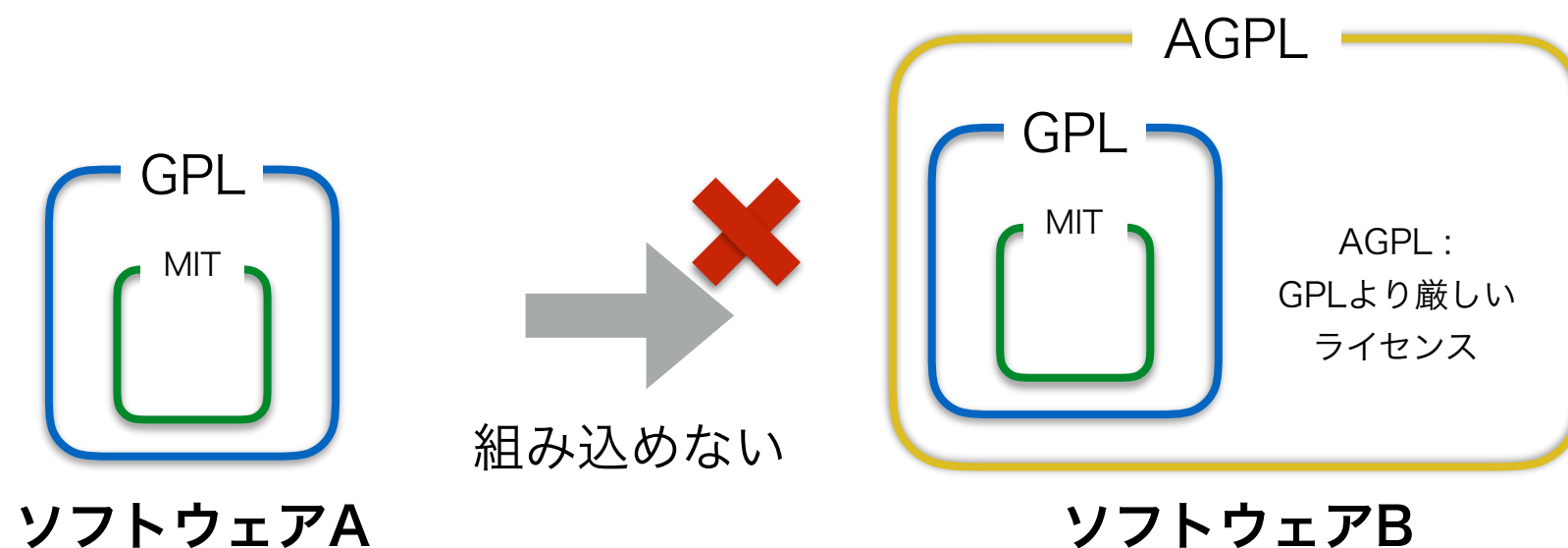
ただし、義務と権利が衝突するため、
列挙した全ての権利を得られるわけではない

ライセンスの衝突

- ✧ あるライセンスによって得られる権利が他のライセンスの義務(obligation)によって得られなくなること

- 例：Sublicenseに関する条項

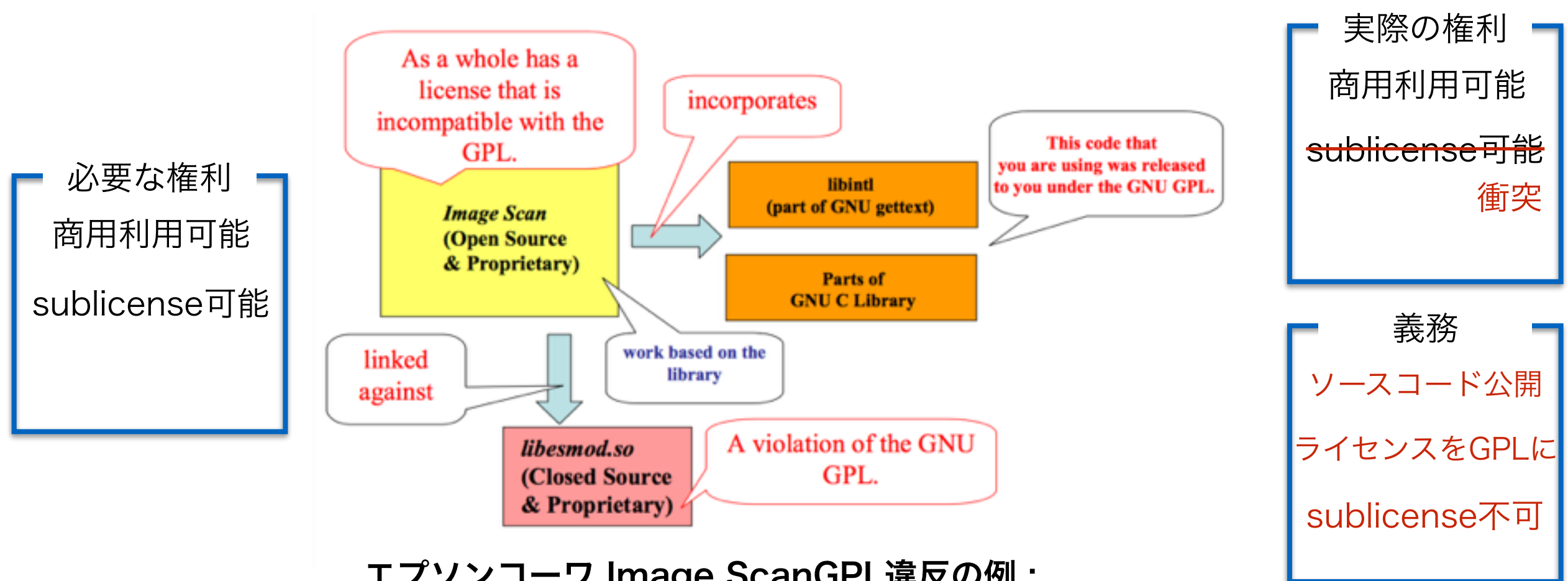
- ★ MITの権利：より厳しいライセンスに組み込むことができる
- ★ GPLの義務：より厳しいライセンスに組みこまない



この場合、MITのsublicenseの権利は得られない

HtLSの設計

- システム設計者は、HtLS全体としての権利と義務を明確化し、必要な権利が得られるようにOSSを選ばなければならない



エプソンコーワ Image ScanGPL違反の例：

GPLが含まれたソフトウェアをエプソンのライセンスとして配布
欲しい権利を得ていなかった上に、義務も果たしていなかった

問題点

- ✧ HtLS設計の際、含まれている全てのライセンスの権利と義務について、その衝突について検証する必要がある
- ✧ ただし、同一システムに多くの異なるライセンスが含まれている場合、ライセンスが衝突するかどうかを判断するには労力がかかる
 - ✧ 単純に数が多い
 - ✧ 衝突の判定は法律素人にはしんどい

研究の目的とアプローチ

- ❖ 目的：コンポーネントの関係とそのライセンスから、ライセンス衝突を計算した上で、システム全体の義務と権利を自動で出力する手法の提案
- ❖ アプローチ
 - ★ ライセンスを権利と義務をタプルで表現したメタモデルに変換
 - ★ メタモデルとシステムアーキテクチャの情報からライセンスの衝突を計算

License metamodel

- ✧ ライセンスを1つ以上の権利と0個以上の義務をタプルで表現する

- ★ Actor : licensor, licensee

Restrict Natural Language Statement Textual Formを参考

- ★ Modality

- ✧ 権利 : May or Need Not

- ✧ 義務 : Must or Must Not

- ★ Action : sublicense, modifyなど

- ★ Object : 特定のソフトウェア名など

- ★ License : このライセンスなのか, 他のライセンスなのか

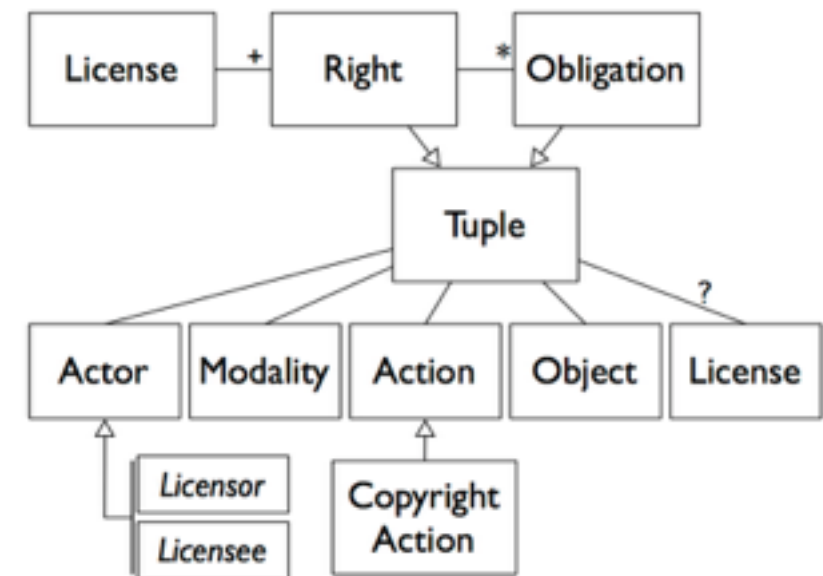


Figure 3. The metamodel for licenses

実例

- ✧ 9つのOSSライセンスを実際に分析し，Modality, Object, Licenseに入りうるものを調査

	Modality	Object	License (optional)
Abstract Right	May or Need Not	Any Under This License	This License or Object's License
		Any Source Under This License	
		Any Component Under This License	
Concrete Right	Must or Must Not	Concrete Object	Concrete License
Concrete Obligation			
Abstract Obligation			
	Must or Must Not	Right's Object	Concrete License or Right's License
		All Sources Of Right's Object	
		X Scope Sources	
		X Scope Components	

Abstract Right : Objectが具体的ではない

-Licensee · may · distribute <Any Source> under <This License>

Concrete Obligation : Objectが具体的

-Licensee · must · retain the [BSD] copyright notice in [file.c]

Figure 4. Modality, object, and license

ライセンス衝突の検出の入出力

✧ 入力

- システムアーキテクチャと各コンポーネントのライセンス情報
- 各ライセンスのライセンスアーキテクチャ

✧ 出力

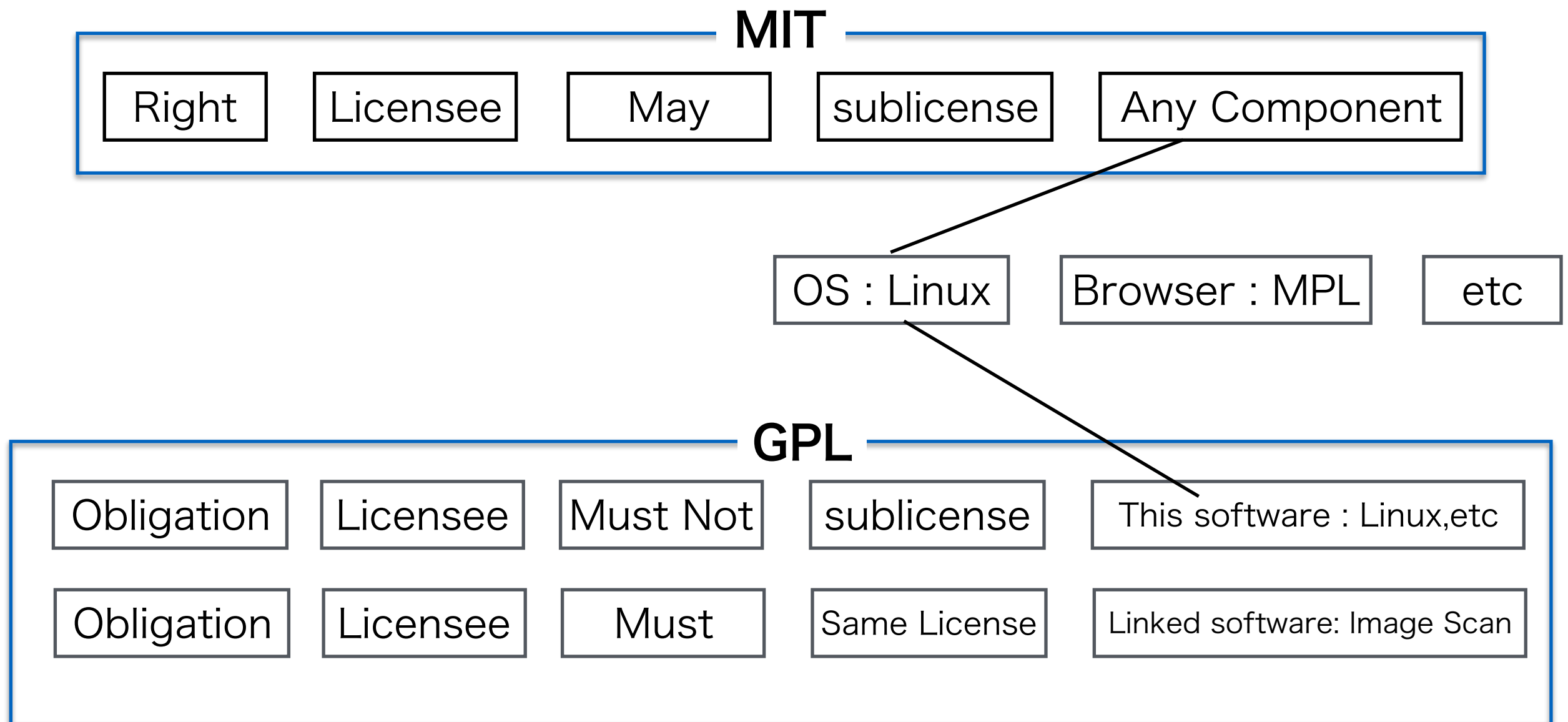
- 得られる権利, 果たすべき義務

ライセンス衝突の検出方法

- ✧ 1.各ライセンスの権利と義務をオブジェクトを介して追跡
 - あるライセンスの権利を得るための義務をライセンスアーキテクチャ集合からトレース
 - あるライセンス義務を果たすことで得られる権利をライセンスアーキテクチャ集合からトレース
- ✧ 2.抽出した権利と義務に衝突が発生しているかどうか

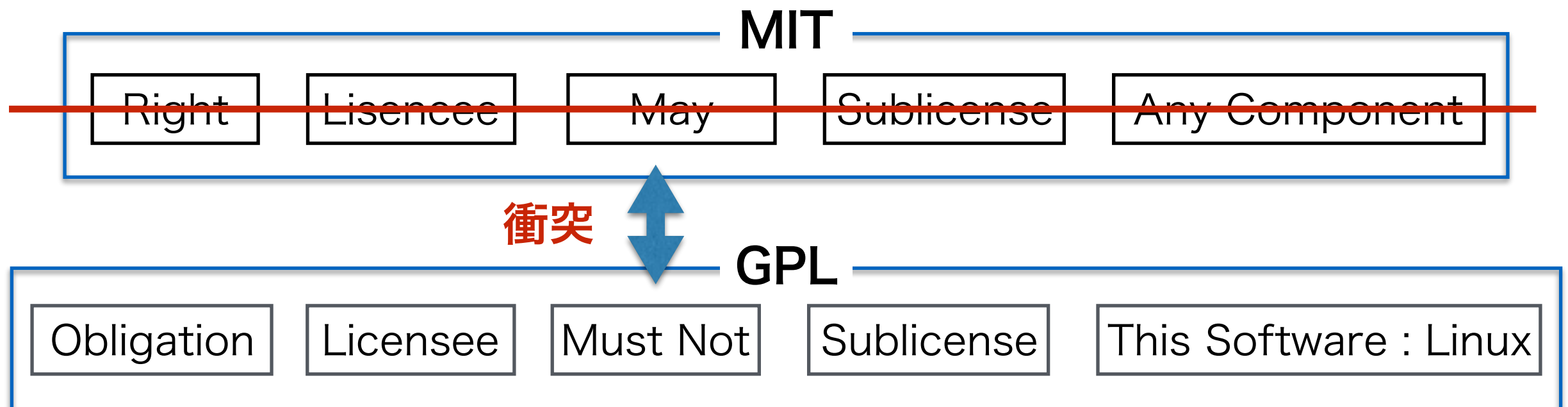
ライセンス衝突の検出方法：Step1

- ＊ あるライセンスの権利を満たすための義務をライセンスアーキテクチャ集合から抽出



ライセンス衝突の検出方法：Step2

- ✧ 抽出した権利と義務に衝突が発生しているかどうか
- ✧ 衝突した状態
 - Actor, Action, Objectが同一
 - Modalityが対になってる
 - may <-> must not, may not <-> must not



GUIにしてみた

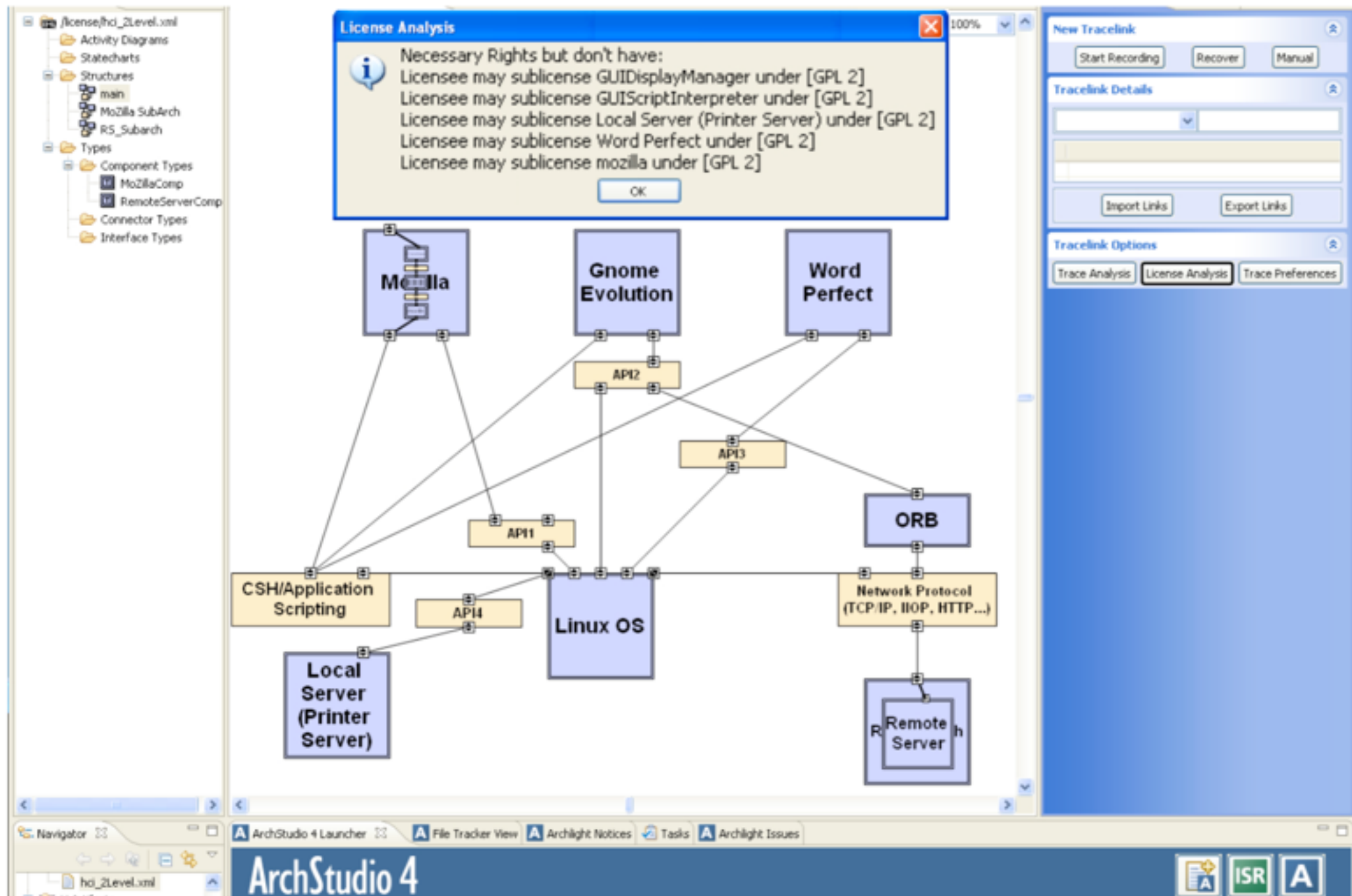


Figure 8. The license architecture analysis tool identifying unavailable rights

利用できない権利を列挙できている

まとめ

✧ まとめ

- HtLSにおけるライセンス衝突の検出するためのライセンスメタモデルを提案した
- 9つのライセンスでは、衝突により利用できない権利を列挙することができた

✧ 今後の課題

- 特許の側面まで手をつけていない
- 選択できるソフトウェアを提示したい