L'ingénierie inverse et la propriété intellectuelle

WOLFGANG STRAUB*

I. Introduction

Il y a quelques décennies, des ingénieurs japonais (et non seulement des Japonais) ont acheté des appareils européens et américains pour les démonter en étudiant soigneusement leur construction. Ils se sont ainsi inspirés du savoir-faire technique caché dans les produits afin de créer des appareils concurrents. Toutefois, seule une partie du savoir-faire était incorporée dans les produits, d'autres éléments se cachant dans les outils de fabrication et les tours de main des collaborateurs. Les «désassembleurs» ont d'abord bénéficié de coûts de production moins élevés que les fabricants des produits originaux. Néanmoins, pour que ces produits restent compétitifs à la longue, il ne suffisait pas de copier afin de rendre au mieux la qualité originale. Il fallait créer . . . Mais que faire si les concurrents les désassemblaient de la même manière?

Suite à l'invention de l'ordinateur et des logiciels, la problématique de la contrefaçon a atteint une nouvelle dimension: un programme d'ordinateur peut être copié en un nombre infini d'exemplaires tout en conservant la même qualité et cela à moindres frais. Les créateurs de software ont inventé un grand nombre de mécanismes techniques pour protéger leurs produits contre toute reproduction. Ces mécanismes ont souvent été surmontés grâce à des mesures d'ingénierie inverse. Le *reverse engineering* pose donc des problèmes particuliers pour les produits software. C'est la raison pour laquelle on trouve, en droit suisse ainsi que dans les législations étrangères, des dispositions particulières pour la décompilation de logiciels alors, que pour d'autres produits, on cherchera en vain des textes légaux qui se réfèrent explicitement à l'ingénierie inverse.

---

* Dr en droit, LL. M., avocat à Berne. Je tiens à remercier Me Michel Jaccard, dr. en droit. LL. M., M. Yann E. Magnin, LL. M., Me Ralph Schlosser, dr. en droit, LL. M., et M. David von Felten de leurs conseils précieux.

II. L'ingénierie inverse et le droit d'auteur

Le droit d'auteur traditionnel vise la protection d'œuvres littéraires et d'art, donc d'objets sans caractère technique. Bien que ces œuvres puissent cacher un savoir-faire artistique, ce dernier reste inaccessible à des mesures d'ingénierie inverse[1]. Pour des raisons historiques, la protection des logiciels a été prise en compte dans un premier temps exclusivement par le droit d'auteur. Au vu toutefois de leur caractère plutôt technique[2], on a assuré leur protection par le biais d'une fiction légale, les assimilant aux œuvres littéraires[3].

La protection des logiciels par le droit d'auteur, bien qu'elle soit commode et ne présuppose aucune formalité, souffre de certaines faiblesses: la condition de l'individualité[4] entraîne des incertitudes quant à la détermination des élé-

1 Bien que pour des peintures une analyse de différentes couches de pigment puisse servir à reconstruire le processus de création, ce savoir ne permet pas encore de reprendre ses éléments individuels. Par exemple, le film ‹Le mystère de Picasso› par HENRI-GEORGES CLOUZOT ne mettait nullement en danger la propriété intellectuelle du peintre en visualisant sa méthode de travail.

2 Toutefois, le *caractère technique* des logiciels a été fortement contesté dans le droit des brevets d'invention. Voir pour l'évolution historique de la notion de technicité ROBERT G. BRINER, Informatik-Erfindungen in: INGRES (éd.) Kernprobleme des Patentrechts; Festschrift zum einhundertjährigen Bestehen eines eidgenössischen Patentgesetzes, Zurich 1988, p. 179 ss, p. 187 ss, et pour le droit communautaire ANSGAR OHLY, Software und Geschäftsmethoden im Patentrecht, CR 2001, p. 809 ss, p. 810 ss.

3 Art. 2 al. 3 LDA. Selon l'art. 10 ADPIC, les programmes d'ordinateur, qu'ils soient exprimés en code source ou en code objet, seront protégés en tant qu'œuvres littéraires en vertu de la Convention de Berne pour la protection des œuvres littéraires et artistiques. Voir aussi l'art. 4 du traité de l'OMPI sur le droit d'auteur du 20 décembre 1996. L'avant-projet de l'IFPI pour une révision du droit d'auteur précise donc dans son art. 2 al. 3 que les programmes d'ordinateurs (logiciels) sont protégés en tant qu'œuvres littéraires quel qu'en soit le mode ou la forme d'expression.

4 Malgré la mention particulière des programmes d'ordinateurs à l'al. 3 de l'art. 2 LDA, la doctrine suisse est majoritairement d'avis que les programmes ne sont protégés que s'ils répondent aux exigences de l'*individualité*. Ceci présuppose que l'auteur profite de façon individuelle d'une marge de créativité. Voir NEFF/ARN, SIWR II/2, p. 138 ss. Voir par contre BARRELET/EGLOFF, Le nouveau droit d'auteur, 2e éd., Berne 2000, n. 25 ad art. 2 LDA. L'art 1 al. 3 de la directive 91/250 CE du 14 mai 1991 concernant la protection juridique des programmes d'ordinateurs (JOCE L 122/42 du 17 mai 1991) semble mettre sur le même plan les critères de la création intellectuelle et de l'individualité. Voir WOLFGANG STRAUB, Der Sourcecode von Computerprogrammen im schweizerischen Recht und der EU-Richtlinie über den Rechtsschutz von Computerprogrammen, UFITA 2001 III, p. 807 ss, cit. Straub, Sourcecode, p. 811 ss, avec des références.

ments à protéger[5] et de l'étendue de la protection[6]. Entre-temps, une tendance internationale[7] veut ouvrir aux logiciels la protection du *droit des brevets*[8].

Les logiciels ont introduit la problématique de l'ingénierie inverse dans le domaine du droit d'auteur. Par ailleurs, des produits multimédia (p. ex. DVD vidéo/jeux interactifs) ont assoupli les limites traditionnelles entre software et œuvres artistiques et littéraires au sens propre du terme. Des œuvres artistiques enregistrées sous forme numérique[9] peuvent aussi être désassemblées[10]. Pour des produits hybrides du genre multimédia, les dispositions particulières sur l'analyse et la décompilation de logiciels ne sont applicables qu'aux composants software.

---

5 Du point de vue du droit d'auteur, non seulement des séquences d'instructions, mais aussi la *structure d'un programme* peuvent être protégées. Par contre, la définition de la tâche à laquelle le logiciel répond est exclue de la protection. Selon le message, FF 1989 III, p. 523, les algorithmes à la base d'un programme d'ordinateur ne peuvent pas être protégés. La question de savoir dans quelle mesure la forme protégée par la loi sur le droit d'auteur peut être délimitée des *idées* non susceptibles d'être protégées reste toutefois controversée. Voir IVAN CHERPILLOD, L'objet du droit d'auteur, Etude critique de la distinction entre forme et idée, thèse Lausanne 1985, cit. Cherpillod, L'objet du droit d'auteur, p. 59 ss et 171 ss.

6 Voir WOLFGANG STRAUB, Individualität als Schlüsselkriterium des Urheberrechts, GRUR Int. 2001, p. 1 ss, p. 7. D'ailleurs, le droit d'auteur ne vise pas à interdire des créations parallèles d'une œuvre pourvu qu'elles soient élaborés de manière indépendante *(rarissime!)*.

7 Au niveau européen, l'article 52 al. 2 lit. c CBE n'admet pas le *dépôt de brevets* sur des programmes d'ordinateurs entiers. Une première tentative d'éliminer cette disposition a échoué. Toutefois, son alinéa 3 limite l'exclusion au dépôt de logiciels *tels quels*. Depuis la décision de la Chambre de recours technique 3.5.1 en date du 1er juillet 1998, T 1173/97, JO OEB 1999, p. 609 ss (IBM produit ‹programme d'ordinateur›) les conditions de protection ont été libéralisées. Voir MARGARETE SINGER/DIETER STAUDER, Europäisches Patentübereinkommen, 2e éd., Cologne/Berlin/Bonn/Munich 2000, n. 36 ss ad art. 52 CBE; ALOIS FRAUENKNECHT, Patente – Quo vadis? Antworten und Konsequenzen, SIC 2001 p. 715 ss, p. 717; ANTHONY HOWARD, Patentability of Computer-Implemented Inventions, A concise analysis of the Commission's proposal for a Directive on the patentability of computer-implemented inventions, CRi 2002, p. 97 ss; ANDREAS WELCH/CHRISTOPH MÜLLER, Patente – Quo vadis? – Eine Erwiderung, sic 2002, p. 290, et OHLY (note 2), p. 810 ss, se référant aussi à l'art. 27 ADPIC. Voir aussi la proposition de la Commission européenne du 20.2.2002 pour une directive concernant la brevetabilité des inventions mises en œuvre par ordinateur, COM (2002) 92.

8 L'IFPI tente d'appliquer la loi sur les brevets d'invention conformément à la CBE, sans pour autant marquer un changement fondamental après la décision de l'OEB dans l'affaire T 1173/97. Voir aussi PETER HEINRICH, Kommentar zum Schweizerischen Patentgesetz und den entsprechenden Bestimmungen des Europäischen Patentübereinkommens, Zurich 1998, n. 1.17 ss ad art. 1 LBI, et ch. 223.4 des directives pour l'examen quant au fond. Voir pour l'évolution historique de la protection des logiciels BRINER (note 2), p. 187 ss, et ALEXANDRA FREI, Softwareschutz durch das Patentrecht in: Felix H. Thomann/Georg Rauber, Softwareschutz, Berne 1998, p. 98 ss.

9 Même pour des enregistrements analogues, des questions similaires peuvent se poser. Voir la décision de la 2e chambre pénale du tribunal cantonal de Zurich du 16.1.1996 (Walt Disney), RSPI 1996, déclarant licite la conversion d'un film vidéo d'une norme technique en une autre.

10 Contrairement aux programmes d'ordinateurs, l'application de l'art. 19 LDA n'est pas exclue.

1. Code source et code objet

Les logiciels sont développés en différentes étapes qui dépendent de la taille du projet. En principe, le développement commence par un plan de la structure du programme d'ordinateur. Ce plan forme, avec d'autres informations, la *documentation pour le développement*. Cependant, une telle documentation particulière est souvent remplacée par des remarques de l'analyste programmeur dans le code source.

Les ordinateurs ne peuvent comprendre les programmes que s'ils sont constitués en forme numérique, c'est-à-dire une suite de chiffres binaires[11]. Le *code source* d'un programme est mis en œuvre en écrivant les instructions, commande par commande, ligne par ligne, dans un langage de programmation et en insérant des références à des sous-programmes et à des bibliothèques de programmation (DLLs). Des étapes supplémentaires sont encore nécessaires pour le rendre utilisable par l'ordinateur: avec des logiciels *compiler*[12] et *linker*[13], les commandes doivent ainsi être transformées en *code objet*[14].

---

11 Ainsi conçue, la programmation serait toutefois peu efficace, car chaque commande consiste en une séquence embrouillée formée de chiffres ‹un› et ‹zéro›. C'est pourquoi les logiciels sont écrits en *langages de programmation – supérieurs –*, permettant une combinaison logique des commandes. L'informatique distingue entre les langages de programmation du genre *Assembler* et les *langages* – supérieurs (par exemple Basic, C, Pascal, Modula 2, Java etc.). Les langages Assembler sont bien adaptés aux composants hardware, ce qui facilite leur transformation en code numérique (*executable*). Ils permettent de formuler avec grande précision les instructions mais leur utilisation est difficile. La plupart des programmes d'ordinateurs sont ainsi écrits dans des langages supérieurs, mieux adaptés à la pensée humaine et à la définition des tâches particulières.

12 Au lieu d'une transformation unique du code source en code objet, des *programmes d'interprétation* peuvent être utilisés pour effectuer une traduction simultanée (par exemple, le langage de programmation Java s'appuie sur ce principe). Pourtant, la vitesse d'exécution du logiciel se réduit par un tel procédé.

13 Le plus souvent, les logiciels consistent en divers sous-programmes, n'étant pas nécessairement disponibles sous forme de code source. Quand les interfaces sont connues, même des programmes en code objet peuvent être incorporés. Par un *linker*, les différents ‹objets› sont liés entre eux ainsi qu'à des bibliothèques de programmes (DLL) et enregistrés dans un cadre permettant de les charger dans la mémoire vive. La compilation et le *linking* ne sont pas forcément des étapes visiblement différentes pour l'analyste-programmeur.

14 L'informatique opère une distinction entre le *code objet* (des ‹objets› sont des programmes d'ordinateurs compilés, pas encore compréhensibles pour le matériel hardware mais plus aisés à décrypter pour l'homme) et le code numérique (*binary code* ou *executable*) consistant en informations numériques liées par un programme linker. Pour ne pas augmenter la confusion terminologique, l'expression ‹code objet› est ici utilisée conformément à la terminologie juridique, comme visant non seulement le code objet, mais aussi le code numérique.

Le programme accompli et le système d'exploitation créent ensemble une *interface d'utilisateur* (par exemple interface Windows) par le biais de laquelle l'utilisateur peut enfin communiquer avec le logiciel.

2. *Possibilités et limites de la décompilation*

Puisque pour l'utilisation d'un logiciel sur un ordinateur, seul le code objet est nécessaire, la plupart des logiciels sont fournis sous cette forme. Cependant, pour la modification, la création de logiciels interopérables, des analyses de sécurité etc., il peut se révéler nécessaire de disposer de certaines informations uniquement contenues dans le code source.

Il existe différentes possibilités de récupérer de telles informations à partir du code objet. On peut distinguer les méthodes d'analyse sans transformation du code objet (par exemple le *line tracing*[15]) et la décompilation[16]. Toutefois, la reconstruction du code source n'est pas seulement exigeante d'un point de vue technique, mais encore elle ne sera jamais identique avec le code original[17].

3. *L'intérêt de l'auteur à la confidentialité*

Les programmes d'ordinateur consistent d'ordinaire non seulement en parties protégées, mais encore en passages qui peuvent contenir un savoir-faire précieux sans pour autant bénéficier de la protection de la loi sur le droit d'auteur. Comme il est toutefois difficile de discerner les *parties protégées,* au sens du droit d'auteur, des incertitudes subsistent quant à la libre utilisation de telle ou telle séquence du code source.

---

15 Pour le *line tracing*, le code objet est exécuté pas à pas par un logiciel auxiliaire, en observant d'une façon continue l'effet de chaque commande sur la mémoire vive.

16 Au sujet des différentes méthodes de *software reverse engineering* et leurs aspects juridiques, voir Jochen Marly, Urheberrechtsschutz für Computersoftware in der Europäischen Union, Abschied vom überkommenen Urheberrechtsverständnis, Munich 1995, p. 269 ss, et Thomas C. Vinje, Die EG-Richtlinie zum Schutz von Computerprogrammen und die Frage der Interoperabilität, GRUR Int. 1992, p. 250 ss, p. 253 s.

17 Les informations éliminées par la compilation (par exemple les remarques de l'analyste programmeur sur la structure du logiciel) sont perdues à jamais lors de la transformation en code objet. Le code source ne peut pas être reconstitué dans le langage de programmation original, mais seulement en *Assembler*. Des recherches statistiques ont révélé que seuls les 75–98% environ du code source original peuvent être décompilés. Voir Marly (note 16), p. 275. Cependant, une reconstruction complète n'est souvent pas nécessaire pour des modifications ou la création de logiciels interopérables.

Une décompilation accroît le risque d'infractions à la loi sur le droit d'auteur[18], mais elle entraîne aussi des *conséquences économiques* pour l'auteur[19]: par l'emprunt de parties non individuelles du logiciel et de savoir-faire compris dans le code source, les compétiteurs peuvent bénéficier d'avantages par rapport à la concurrence. En outre, la découverte du code source permet souvent à l'acquéreur d'un logiciel de le modifier sans l'aide du producteur[20].

4. Les intérêts de l'acquéreur

Bien que, pour l'exécution d'un logiciel par un ordinateur, seul le code objet soit nécessaire, l'acquéreur peut avoir des intérêts légitimes à disposer du code source puisqu'il est le plus souvent indispensable pour des *modifications* (par exemple correction d'erreurs ou adaptation à de nouveaux systèmes informatiques)[21], des *analyses de sécurité,* etc.

En cas de doute sur la capacité du fournisseur à développer de façon continue un logiciel coûteux (surtout en cas de logiciels individuellement développés), la disponibilité du code source donne une certaine garantie à l'acquéreur[22].

5. L'importance des interfaces

Les programmes d'ordinateurs doivent communiquer et fonctionner de manière interopérable avec des composants hardware et software provenant d'une multitude de producteurs. Par exemple, la banque de données d'une entreprise n'est utile que si elle peut être utilisée en corrélation avec d'autres applications (des logiciels de traitement de texte, de comptabilité, etc.). Pour que des programmes d'ordinateur puissent importer des informations d'un autre

---

18 Le code objet décompilé permet par exemple d'effacer le numéro de série ou le nom du producteur pour fabriquer des copies pirates. Il est même possible de réarranger un logiciel de telle façon que le piratage soit difficile à prouver. Voir la décision du Tribunal cantonal de Zurich du 13.11.1993 (Netware *serial updating program*), RSPI 1994, p. 181 ss.

19 Par le terme d'*auteur* on vise ici tout titulaire d'un droit d'auteur (voir aussi l'art. 17 LDA).

20 Voir cependant pour l'admissibilité de modifications ATF 125 III 262 et Christine Chappuis, Note sur le devoir d'information du donneur de licence à l'égard du preneur de licence en matière informatique, SJ 1999, p. 474 ss.

21 Voir sur l'admissibilité de *modifications* infra note 71.

22 Pour tenir compte des intérêts des deux parties dans de telles situations, des contrats d'*escrow*, prévoyant le dépôt du code source auprès d'un tiers avec obligation de remise à l'acquéreur dans des situations déterminées (par exemple la faillite du déposant), peuvent être conclus. Voir Kurt U. Blickenstorfer, Der Sourcecode-Escrow, in: Softwareschutz, édité par Felix H. Thomann/Georg Rauber, Berne 1998, p. 212 ss, et Stefan Gerster, Das Escrow Agreement als obligationenrechtlicher Vertrag, thèse Zurich 1991.

logiciel, certaines informations sont nécessaires (par exemple le format et l'adresse interne du fichier dans lequel les données sont mémorisées). D'ordinaire, les informations nécessaires à l'interopérabilité sont contenues dans le code source, mais elles ne sont pas accessibles par l'interface graphique du programme d'ordinateur[23]. Les interfaces, indispensables pour le développement de logiciels interopérables, peuvent par conséquent avoir une valeur économique et stratégique considérable pour le libre jeu de la concurrence[24].

Enfin, des raisons économiques peuvent aussi favoriser une libre divulgation des interfaces. Un choix de progiciels compatibles augmente la valeur pratique et le champ d'application d'un programme d'ordinateur.

Les interfaces sont souvent déterminées par leur fonction, de sorte qu'il ne reste qu'une marge de créativité très limitée[25]. Si une *interface* est tout de même *protégée,* l'auteur du programme interopérable doit, sous réserve de la ‹libre utilisation›[26], chercher pour son propre logiciel une solution en dehors du domaine protégé[27]. On doit cependant distinguer les *spécifications* (par exemple la forme des données) de l'interface au sens strict.

---

23 Par exemple, les *dates* sont souvent enregistrées sous forme de deux chiffres pour l'année et ensuite liées par le logiciel avec les deux chiffres pour le siècle afin d'être affichées en quatre chiffres. Ce procédé a causé le fameux ‹problème de l'an 2000›, voir WOLFGANG STRAUB, Die Haftung für das Jahr-2000-Problem, PJA 1999, p. 524 ss.

24 La Commission des Communautés européennes a déjà été confrontée à un cas analogue en 1984. IBM ne publiait les interfaces hardware d'un ordinateur de système modulaire qu'après le lancement, ne permettant pas aux concurrents de développer à temps des composantes compatibles. Ce litige se termina par une transaction dans laquelle IBM s'engagea à publier les informations correspondantes à temps. Voir bull. CE 7/8 1984, p. 7 ss et 10 1984, p. 105 ss, et les commentaires de MARLY (note 16), p. 309 ss, et JÉRÔME HUET, Le *reverse engineering*, ou ingénierie inverse et l'accès aux interfaces dans la protection des logiciels en Europe: questions de droit d'auteur et de droit de la concurrence, Recueil Dalloz Sirey, 1990, partie chronique, p. 99 s., cit. Huet, Reverse engineering. Voir pour l'importance du droit des cartels sur le droit d'auteur en général et la question des interfaces en particulier IVAN CHERPILLOD, Droit d'auteur et droit des cartels, medialex 1997, p. 146 ss, cit. Cherpillod, Droit d'auteur, p. 149 ss.

25 Voir pour la condition de l'individualité supra n. 4–6.

26 Le principe de la ‹libre utilisation› (freie Benutzung) qui permet l'utilisation d'éléments protégés d'une œuvre préexistante si leur importance s'efface par rapport à l'individualité, est désormais reconnu par le Tribunal fédéral (voir ATF 125 III 328 consid. 4 c). Voir pour l'usage libre CHERPILLOD, L'objet du droit d'auteur (note 5), p. 145 ss, et IDEM, SIWR II/1, p. 277 s.; BARRELET/EGLOFF (note 4), n. 12 ad art. 11 LDA, et KAMEN TROLLER Manuel du droit suisse des biens immatériels, 2e éd., Bâle/Francfort-sur-le Main 1996, tome I, p. 259. *Contra*: FRANÇOIS DESSEMONTET, Le droit d'auteur, Lausanne 1999, note 400 ss.

27 Au niveau communautaire, un premier projet de directive, excluant les interfaces de toute protection du droit d'auteur, n'a pas été adopté. Voir JÉRÔME HUET, L'Europe des logiciels: les droits des utilisateurs, Recueil Dalloz, partie chronique, 1992, cit. Huet, L'Europe des logiciels, p. 315 ss. Si la réalisation d'une interface ne se base que sur des nécessités techniques, elle peut librement être copiée puisqu'elle manque d'individualité. Voir aussi MARLY (note 16), p. 324, VINJE (note 16), p. 259 s., et BRIDGET CZARNOTA/ROBERT J. HART, Legal Protection of Computer Programs in Europe – A Guide to the EC Directive, Londres/Dublin/Edinburgh/Munich 1991, p. 82.

6. *L'insuffisance du droit d'observation*

D'un point de vue technique, toute utilisation d'un logiciel affecte le droit d'auteur puisqu'elle entraîne une *copie éphémère dans la mémoire vive* de l'ordinateur[28]. Ainsi, l'observation du fonctionnement d'un programme nécessite par principe une licence[29] de duplication[30]. La reconstruction du code source à partir du code objet présuppose en plus une licence de modification[31]. Par analogie à la directive 91/250 CE, la loi suisse sur le droit d'auteur contient des *licences légales* pour l'observation du fonctionnement d'un programme sans modification du code[32] et pour la reconstruction du code source par le biais de la décompilation[33].

Le droit d'observation repose sur le principe que seule *l'expression*, et non pas les *idées* qui sont à la base d'une œuvre, peut être protégée par le droit d'auteur. L'utilisation comprend le *droit d'observer, d'étudier et de tester le fonctionnement d'un programme* afin de déterminer les idées et les principes qui sont à la base de ses éléments[34]. L'interface graphique du logiciel et les résultats de l'utilisation ne permettent presque jamais la déduction des idées puisque les mêmes résultats peuvent être obtenus par des logiciels de structures fondamentalement différentes.

Une analyse de logiciels n'est permise que dans les limites très restreintes de l'art. 17 ODAu: d'une part, le droit d'analyse ne s'étend pas à la structure du programme entier, mais il est limité à des *éléments particuliers du logi-*

---

28 Voir Neff/Arn, SIWR II/2, p. 227 s., et Ursula Widmer, Der urheberrechtliche Schutz von Computerprogrammen, RDS 1993 I, p. 247 ss, p. 261.

29 Voir pour la *nature juridique* d'une telle licence Reto M. Hilty, Lizenzvertragsrecht, Bern 2001, cit. Hilty, Lizenzvertragsrecht, p. 262 s.

30 Voir aussi l'art. 4 lit. a de la directive 91/250 CE, mettant au clair que toute reproduction permanente ou provisoire d'un programme d'ordinateur, en tout ou en partie, par quelque moyen et sous quelque forme que ce soit relève du droit d'auteur. Voir pour la congruence de cette disposition avec le droit suisse Reto M. Hilty, Der Schutz von Computerprogrammen – nationale und internationale Normen auf dem Prüfstand des Internets, SIC 1997, p. 128 ss, cit. Hilty, Computerprogramme, p. 136.

31 Voir Georg Rauber, Der urheberrechtliche Schutz von Computerprogrammen; eine problemorientierte Untersuchung des Werkbegriffs nach schweizerischem und internationalem Urheberrecht, Zurich 1988, thèse Zurich 1988, p. 255, et IDEM, Inhalt und Schranken des urheberrechtlichen Softwareschutzes in: Software-Schutz Software-Haftung, Zurich 1992, p 33 ss, p. 40 s.

32 Art. 17 al. 1 ODAu et art. 5 al. 3 de la directive 91/250 CE.

33 Art. 21 LDA et 17 al. 2 ODAu et art. 6 lit. c de la directive 91/250 CE.

34 Voir l'art. 17 al. 1 ODAu. Le législateur suisse a manqué de retenir la *nature contraignante* du droit d'observation. Voir Gianni Fröhlich-Bleuler, Urheberrechtliche Nutzungsbefugnisse des EDV-Anwenders, PJA 1995, p. 569 ss, p. 577. La directive 91/250 opère une distinction entre le droit contraignant de pure observation et le droit dispositif d'analyse (art. 4 et art. 9 al. 1).

ciel[35]. En plus, l'analyse ne doit pas dépasser le cadre des actes conformes d'utilisation du logiciel[36]. Toutefois, le chargement, l'affichage, le passage, la transmission ou le stockage du programme[37] ne permettent l'analyse d'un logiciel que dans des limites le plus souvent insuffisantes pour des modifications efficaces. Une analyse complète n'est possible que par décompilation.

7. Le régime suisse de la décompilation des interfaces

La décompilation, créant une copie du code objet sous une forme différente présuppose une licence correspondante. La directive 91/250 CE et la loi suisse sur le droit d'auteur ne traitent que de la décompilation des interfaces[38] pour la création de logiciels interopérables[39]. L'art. 21 LDA a été inséré par le Conseil national sous l'influence de la *directive 91/250 CE*[40]. Le texte de cette disposition[41] a cependant été simplifié par rapport à la formule complexe de la directive[42].

---

35 Selon Marly (note 16), p. 272, et Czarnota/Hart (note 27), p. 70, on doit distinguer entre l'observation licite du fonctionnement et l'analyse non licite du logiciel même.

36 Neff/Arn, SIWR II/2, p. 256, et pour le droit communautaire Czarnota/Hart (note 27), p. 69, partent de l'idée que les techniques nécessitant des modifications du code objet sont exclues.

37 Art. 17 al. 1 lit. a ODAu. Voir aussi l'art. 5 al. 3 de la directive 91/250 CE.

38 Le terme d'*interface* comprend des séquences de code source indispensables pour l'échange des données entre programmes ou sous-programmes d'ordinateurs (objets). Au-delà des interfaces dans ce sens strict, des informations supplémentaires non contenues dans ces parties du code source peuvent être nécessaires à l'interopérabilité. Une décompilation ne présuppose pas que les interfaces soient déjà conçues pour l'échange de données avec des logiciels interopérables (interfaces ouvertes). Voir par contre Xavier Linant de Bellefonds, Le droit de décompilation des logiciels: une aubaine pour les cloneurs? La Semaine Juridique I/1998, p. 479 ss, p. 480 s.

39 Voir art. 21 LDA et art. 17 al. 2 ODAu.

40 Voir pour la genèse de cette disposition Carl Baudenbacher/Irene Klauer, Zur Europafähigkeit der URG-Reform, St. Gall/Berlin 1991, p. 65 ss.

41 Pour la *relation entre l'art. 17 LDA et la directive 91/250 CE*, voir Carlo Govoni, Der urheberrechtliche Schutz von Computerprogrammen, PJA 1993, p. 569 ss; Felix H. Thomann, Die Euroverträglichkeit der Softwareschutz-Regelung gemäss dem revidierten URG, PJA 1993, p. 563 ss; Felix Addor/Carlo Govoni, Die Harmonisierung des europäischen Urheberrechts aus schweizerischer Perspektive, ZUM 1995, p. 464 ss, p. 465 s., et Neff/Arn SIWR II/2, p. 39 s. Voir aussi l'art. 13a al. 2 de l'avant-projet de l'IFPI pour la révision du LDA, se rapprochant encore plus du libellé de la directive 91/250 CE.

42 Les dispositions sur la décompilation étaient très controversées lors des débats sur la directive 91/250 CE. Le régime actuel est le résultat d'un compromis entre les intérêts des fabricants de logiciels de base et les utilisateurs et producteurs de logiciels interopérables. Voir au sujet de *l'histoire de la directive*, Marly (note 16), p. 311 ss; Czarnota/Hart (note 27), p. 7 ss; Huet, l'Europe des logiciels (note 27), p. 315 ss; Michael Lehmann, Die Europäische Richtlinie über den Schutz von Computerprogrammen, GRUR Int. 1991, p. 327 ss; Josette Beer-Gabel/Régis Chemain, La décompilation des logiciels: l'industrie européenne face au droit d'auteur, Revue trimestrielle de droit européen, p. 363 ss, et Robert J. Hart, Interfaces, Interoperability and Maintenance, European Intellectual Property Review 1991, p. 111 ss.

La décompilation présuppose d'abord un *droit d'utilisation* du programme d'ordinateur, découlant d'une cession ou d'une licence[43]. L'art. 21 LDA protège ainsi les intérêts des utilisateurs du logiciel et non ceux des compétiteurs. En effet, ces derniers doivent d'abord acquérir le droit d'utilisation d'un programme avant de pouvoir créer des produits interopérables[44].

Le droit d'auteur permet seulement la décompilation des *informations*[45] *nécessaires*[46] à la création de logiciels interopérables. Il peut se révéler très difficile de les retrouver dans un code volumineux. Même s'il est possible de localiser l'interface cherchée, celle-ci ne contient pas forcément toutes les informations nécessaires à l'interopérabilité (par exemple les commentaires sur les rapports avec d'autres éléments du logiciel manquent). Dans certains cas, une *analyse du programme allant au-delà d'une interface particulière* est indispensable pour garantir l'interopérabilité[47].

Contrairement à la directive 91/250 CE, la loi sur le droit d'auteur remplace l'expression de ‹décompilation› par *›décryptage‹* (Entschlüsselung)[48]. La décompilation ne suffit pas toujours pour récupérer un code utilisable. D'ailleurs, quelques fabricants utilisent des techniques cryptographiques pour rendre la décompilation moins facile. Même à supposer que la Suisse

---

43 Voir art. 12 al. 2 LDA.

44 L'ayant droit ne doit pas nécessairement décompiler lui-même. Il peut *mandater un tiers* pour que celui-ci procède à la décompilation. Le droit de décompilation ne peut toutefois être transféré indépendamment du droit d'utilisation. Voir cependant pour le droit communautaire VINJE, (note 16), p. 254.

45 Il ne faut pas confondre la nécessité de disposer de certaines *informations* relatives à une interface concrète avec la nécessité d'utiliser telle ou telle *interface* d'un logiciel. Bien que certaines connections souhaitables ne soient pas nécessaires à l'interopérabilité, l'auteur d'un programme d'ordinateur dispose d'une *marge d'appréciation* par rapport aux interfaces qu'il veut utiliser aussi longtemps que le programme interopérable reste élaboré de manière indépendante.

46 L'art. 6 al. 1 de la directive 91/250 CE permet la décompilation aussi longtemps qu'elle est *indispensable* (unerlässlich). La version allemande et italienne de l'art. 21 LDA parle des informations nécessaires (erforderlich/informazioni necessarie per l'interfaccia), tandis que le texte français mentionne les ‹informations sur des interfaces› sans autre précision. Le critère du caractère nécessaire de la décompilation est toutefois repris dans toutes les langues à l'art. 17 ODAu: ‹. . . les informations nécessaires sur les interfaces sont celles qui sont indispensables à l'élaboration de l'interopérabilité›.

47 Voir CZARNOTA/HART (note 27), p. 77; VINJE, (note 16), p. 257, et ALAIN BENSOUSSAN, le logiciel et le droit, 2e éd., Paris 1998, n. 11 323; *contra* LINANT DE BELLEFONDS (note 38), p. 482. Si l'étendue de la décompilation nécessaire n'est pas motivée par les intérêts de l'acquéreur, l'art. 21 al. 2 LDA contient une ‹soupape de sûreté›.

48 Voir aussi le libellé de l'art. 21 de l'avant-projet de l'IFPI pour une révision du droit d'auteur, utilisant le terme de ‹traduction de la forme du code› calqué sur l'art. 6 al. 1 de la directive 91/250 et l'art. L 122–6-1 IV du Code français de la propriété intellectuelle. Selon BENSOUSSAN (note 47), n. 11 310, cette notion s'avère plus large que la pure décompilation.

opte bientôt pour une protection pénale des mesures techniques[49], l'emploi d'autres techniques nécessaires au décryptage reste licite aussi longtemps que la décompilation est admise.

Une décompilation n'est pas nécessaire si les informations relatives à l'interopérabilité sont déjà accessibles. Faut-il donc contacter le producteur du programme avant de passer à l'ingénierie inverse? Selon l'art. 17 ODAu, les informations doivent être *librement accessibles*[50]. On ne peut donc pas exiger de l'auteur d'un logiciel interopérable d'aller à la recherche de l'auteur du programme de base[51] ou de se tourner vers d'autres sources[52]. Sous réserve du droit de la concurrence[53], le producteur du programme de base n'est pas obligé de fournir les informations nécessaires à l'interopérabilité, mais s'il ne les met pas librement à disposition, il risque la décompilation. Il ne doit pas demander une *rémunération* pour la valeur de l'interface, mais à la limite une contribution pour les frais du transfert d'informations[54]. L'auteur du logiciel interopérable peut cependant être disposé à payer beaucoup plus afin d'éviter les inconvénients de l'ingénierie inverse et obtenir une licence d'utilisation pour l'interface entière.

Même dans le cadre restreint admis par le droit d'auteur, l'ingénierie inverse peut permettre de prendre connaissance d'un savoir-faire[55] précieux. L'utilisation des interfaces obtenues par la décompilation n'est permise que pour l'éla-

---

49 Voir l'art. 70a l'avant-projet de l'IFPI pour la révision de la LDA, calqué sur l'art. 11 du traité de l'OMPI sur le droit d'auteur du 20 décembre 1996.

50 Voir aussi art. 6 al. 1 lit. b de la directive 91/250 CE, présupposant que les informations n'ont pas déjà été *facilement et rapidement accessibles*.

51 Toutefois, une obligation de contacter préalablement l'auteur peut résulter d'un contrat de licence. Voir aussi les exemples de clauses-types chez BENSOUSSAN (note 47), n. 11 329.

52 Voir pour le droit communautaire MARLY (note 16), p. 319 note 315, qui se réfère à l'histoire de la directive. Selon lui, l'auteur du logiciel interopérable ne doit pas préalablement *contacter le fournisseur du programme de base*. Voir aussi VINJE (note 16), p. 257 et ALFRED P. MEIJBOOM in: Copyright Software Protection in the EC, édité par Herald D. J. Jongen/Alfred P. Meijboom, Deventer 1993, p. 16. Des raisons pratiques peuvent néanmoins justifier le recours à un tel procédé. Les dépenses de l'ingénierie inverse sont toujours considérables. Une réponse négative du producteur du programme de base prouve d'ailleurs l'inaccessibilité des informations nécessaires.

53 Voir CHERPILLOD, Droit d'auteur (note 24), p. 149 ss.

54 Selon IVAN CHERPILLOD, Protection des logiciels et des bases de données, la révision du droit d'auteur en Suisse, expertise des systèmes d'information RSPI 1993, cit. Cherpillod, Protection des logiciels, p. 49 ss, p. 62, une *rémunération raisonnable* peut être exigée. De même avis pour la directive 91/250 CE, CZARNOTA/HART (note 27), p. 80. MICHAEL LEHMANN, Die Richtlinie über den Schutz von Computerprogrammen, in: Michael Lehmann (éd.), Rechtsschutz und Verwertung von Computerprogrammen, 2e éd., Cologne 1993, p. 23, part cependant de l'idée que l'information doit être mise à disposition gratuitement.

55 Voir RALPH SCHLOSSER, Le contrat de savoir-faire, thèse Lausanne 1996, p. 45 ss.

boration, le maintien et l'utilisation d'autres[56] *programmes d'ordinateur interopérables*. Le but visé par le législateur suisse et par la directive 91/250 CE, à savoir de faciliter l'interopérabilité pour que l'utilisateur puisse librement combiner un système d'information d'éléments modulaires, justifie une interprétation large, comprenant l'élaboration d'éléments hardware interopérables[57].

La loi sur le droit d'auteur[58] fait dépendre l'utilisation des interfaces du fait que ni *l'exploitation normale du logiciel* ni les *intérêts légitimes des auteurs* ne soient menacés. Quel est le sens de cette formule sibylline? Il s'agit d'éviter des distorsions de concurrence causées par des producteurs qui tentent de contourner les dépenses de développement grâce à l'ingénierie inverse. De toute façon, de tels logiciels portent souvent atteinte au droit d'auteur ou ne sont pas ‹développés indépendamment›[59].

Enfin, les informations obtenues par décompilation ne doivent pas être utilisées pour le développement, la production et la commercialisation d'un logiciel dont *l'expression est fondamentalement similaire*[60]. Cette formule, utilisée nulle part ailleurs par le droit d'auteur, suggère qu'il s'agit de l'interface graphique ou des fonctions du logiciel. Cependant, elle se réfère à la directive 91/250 CE, visant les programmes d'une structure similaire[61]. La décompilation reste cependant licite quand il s'agit d'assurer l'échange de données avec des *logiciels concurrents*[62].

Selon la directive 91/250 CE[63], les informations obtenues dans le cadre d'une décompilation licite ne peuvent pas être librement divulguées[64]. Les in-

---

56 La formule de l'art. 17 ODAu, calquée sur l'art. 6 al. 1 de la directive 91/250 CE, permet même la décompilation en vue d'assurer l'interopérabilité avec des programmes tiers, interopérables avec le logiciel analysé. Voir VINJE, (note 16), p. 256 s.

57 Voir pour la directive 91/250 CE MARLY (note 16), p. 322 ss, et LEHMANN (note 42), p. 334; *contraire*: LINANT DE BELLEFONDS (note 38), p. 481; CZARNOTA/HART (note 27), p 85, et BENSOUSSAN (note 47), n. 11 324. Certains logiciels – surtout des systèmes d'exploitation – doivent aussi fonctionner de manière interopérable avec des composantes *hardware*. Les limites entre hardware et software s'effaçant de plus en plus, des logiciels peuvent être incorporés dans des modules *firmware* ou être combinés avec des éléments hardware (par exemple des modules *dongle*).

58 Art. 21 al. 2 LDA. Voir aussi l'art. 6 al. 3 de la directive 91/250 CE.

59 Art. 17 al. 2 ODAu.

60 L'art. 17 al. 3 ODAu, en concrétisant la loi sur le droit d'auteur, se réfère à l'art. 6 al. 2 lit. c de la directive 91/250 CE. Voir aussi l'art 21 al. 2 lit. c de l'avant-projet de l'IFPI pour la révision de la LDA.

61 Voir MARLY (note 16), p. 316 s. De tels logiciels portent souvent déjà atteinte au droit d'auteur.

62 Voir pour le droit communautaire MARLY (note 16), p. 316 s.; CZARNOTA/HART (note 27), p. 82; VINJE (note 16), p. 256 s., et LEHMANN (note 42), p. 334; IDEM (note 54), p. 17; *contra* LINANT DE BELLEFONDS (note 38), p. 481.

63 Art. 6 al. 2 lit. b.

64 L'art. 21 al. 2 lit. b de l'avant-projet de l'IFPI pour la révision du droit d'auteur veut reprendre cette disposition.

formations décompilées ne bénéficient d'un droit d'auteur que si elles franchissent elles-mêmes le seuil de l'individualité. Cependant, l'interdiction de divulgation sert à assurer l'utilisation exclusive des informations obtenues par voie de décompilation pour la création de logiciels interopérables. Il s'agit toutefois d'une protection de secrets étrangère à la conception traditionnelle des droits de la propriété intellectuelle.

8. Décompilation au-delà de l'interopérabilité

L'utilisateur d'un programme d'ordinateur peut avoir des intérêts légitimes à une décompilation au-delà de la création de logiciels interopérables[65]. Par exemple, la recherche de lacunes de sécurité, la réparation de défauts ou la vérification de soupçons de piratage peuvent présupposer une analyse du programme. Est-ce que l'art. 12 al. 2 LDA permet l'ingénierie inverse dans de tels cas ou la disposition de l'art. 21 LDA est-elle exhaustive en matière d'ingénierie inverse? La genèse de la loi suisse sur le droit d'auteur ne permet pas de conclusions claires[66]. Le champ d'application des deux dispositions n'est pourtant pas identique. La décompilation des interfaces est permise indépendamment de l'usage conforme du logiciel. Dès lors, l'art. 21 LDA ne peut que partiellement être considéré comme *lex specialis* par rapport à l'art. 12 al. 2 LDA[67].

Le droit communautaire[68] permet la traduction, l'adaptation, l'arrangement et toute autre *transformation* d'un programme d'ordinateur lorsque ces actes sont nécessaires pour permettre à l'acquéreur légitime de l'utiliser d'une manière conforme à sa destination, y compris pour corriger des erreurs. La question de savoir s'il est permis de procéder à l'ingénierie inverse pour assurer l'utilisation du logiciel[69] reste cependant controversée. L'avant-projet

---

65 Voir pour la décompilation en droit anglo-américain, basée sur le principe de *fair-use,* Sega Enterprises Ldt. vs. Accolade Inc., 977 F2d 1510, 9th Cir. (1992); JOCHEN SCHNEIDER, Handbuch des EDV-Rechts, 2e éd., Cologne 1997, p. 426 ss, et MARLY (note 16), p. 295, avec des références supplémentaires.

66 Voir BOCN 1992, p. 4 ss.

67 Voir cependant MARTIN LUTZ, Der Schutz der Computerprogramme in der Schweiz, GRUR Int. 1993, p. 653 ss, p. 661.

68 Art. 5 al. 1 de la directive 91/250 CE.

69 Selon la position commune du Conseil du 13 décembre 1990 en vue de l'adoption de la directive 91/250 CE, l'ingénierie inverse d'un programme d'ordinateur n'est pas admissible pour sa modification (voir GRUR Int. 1991, p. 548 s.). Selon CZARNOTA/HART (note 27), p. 65, même la correction d'erreurs n'implique pas le droit de recourir à des procédures admises dans le cadre de la décompilation d'interfaces. Voir cependant MARLY (note 16), p. 314 ss; HUET, L'Europe des logiciels (note 27), p. 316 s.; BEER-GABEL/CHEMAIN (note 42), p. 376, et SCHNEIDER (note 65), p. 499 ss.

de l'IFPI pour une révision de la loi sur le droit d'auteur reprend cette disposition du droit communautaire sans préciser son étendue[70].

Dès lors, il s'agit d'examiner les limites de l'utilisation du logiciel conforme à sa destination selon les circonstances du cas concret. Si par exemple l'analyse d'un élément du programme est indispensable pour la réparation de défauts[71] ou pour la découverte de lacunes de sécurité et les informations nécessaires ne sont pas accessibles d'une autre façon, l'ingénierie inverse peut être admise[72] *ultima ratio*[73]. Une analyse motivée par des soucis de piratage ou par pure curiosité n'est en revanche pas couverte par la condition de l'utilisation conforme[74].

Est-ce que le droit de la concurrence peut justifier des mesures d'ingénierie inverse interdites par le droit d'auteur? La loi sur les cartels n'est pas applicable aux effets sur la concurrence qui découlent exclusivement de la législation sur la propriété intellectuelle[75]. Par conséquent, le droit de la concurrence ne peut être invoqué qu'en dehors des dispositions spécifiques de la loi sur le droit d'auteur. Tel pourrait être le cas si des informations contenues dans le code source sont retenues afin d'empêcher le développement d'un marché de maintenance[76].

Enfin, l'auteur d'un logiciel peut donner son *consentement à l'ingénierie inverse* de son programme si par exemple le code source a disparu. Puisqu'un droit de décompilation contractuel allant au-delà[77] de l'art. 12 al. 2 LDA est rarissime, une interprétation d'un contrat permettant l'ingénierie inverse n'entre en considération que dans des cas exceptionnels. Si l'auteur ne s'acquitte pas d'une éventuelle *obligation contractuelle de restitution* du code

---

70 Art. 13a al. 1 lit. b de l'avant-projet. Selon le rapport explicatif de l'IFPI, les articles 13a et 21 de l'avant projet ne contiennent cependant aucune modification matérielle du droit actuel.

71 Voir pour l'admissibilité de *modifications* dans le cadre de la réparation de défauts Melchior Caduff, Die urheberrechtlichen Konsequenzen der Veräusserung von Computerprogrammen, thèse Berne 1996, Berne 1997, p. 102 ss; Ivan Cherpillod, SIWRII/1, p. 231 ss; Martin J. Lutz, Les programmes d'ordinateur, in: Fabio Marchetto (éd.), La nouvelle loi fédérale sur le droit d'auteur, Lausanne 1994, p. 167 ss, p. 661, et Straub, Sourcecode (note 4), p. 820 ss.

72 Voir pour le droit communautaire Marly (note 16), p. 314 ss; Huet, L'Europe des logiciels (note 27), p. 316 s.; Beer-Gabel/Chemain (note 42), p. 376, et Schneider (note 65), p. 499 ss.

73 Les limites de l'art. 9 al. 2 de la Convention de Berne doivent toutefois être respectés: Les Etats membres peuvent permettre la reproduction d'œuvres protégées dans certains cas spéciaux, pourvu qu'une telle reproduction ne porte pas atteinte à l'exploitation normale de l'œuvre ni ne cause un préjudice injustifié aux intérêts légitimes de l'auteur.

74 Voir pour le droit allemand Helmut Haberstumpf, Der urheberrechtliche Schutz, in: Michael Lehmann (éd.), Rechtsschutz und Verwertung von Computerprogrammen, 2e éd., Cologne 1993, p. 161.

75 Art. 3 al. 2 LCart. Voir Cherpillod, Droit d'auteur (note 24), p. 146 ss.

76 Voir Cherpillod, Droit d'auteur (note 24), p. 147 ss.

77 Contrairement au droit communautaire (art. 9 al. 1 de la directive 91/250 CE), le législateur suisse a manqué de retenir la *nature contraignante* du droit de décompilation.

source, des mesures d'ingénierie inverse ne peuvent pas directement s'appuyer sur l'art. 98 CO[78].

III. Le droit des brevets

Puisque l'objet d'un brevet est publié, des analyses relatives à des produits brevetés ont pour but de découvrir davantage le savoir-faire non breveté ou le mode le plus efficace de son exécution.

Le brevet confère à son titulaire le droit exclusif d'utiliser *l'invention* professionnellement[79]. L'utilisation d'un *produit* breveté mis en circulation en Suisse avec son consentement ne peut cependant pas être interdit puisque le droit a été épuisé pour un tel produit[80]. Est-ce que l'ingénierie inverse affecte l'objet du brevet ou seulement le produit? La réponse doit varier selon les mesures nécessaires pour l'analyse. Par exemple, la décompilation d'un programme d'ordinateur engendre une reproduction. Si certains de ses éléments sont brevetés, une telle mesure peut tomber à la fois sous le coup de la loi sur les brevets d'invention et de la loi sur le droit d'auteur.

Même si l'ingénierie inverse d'un produit breveté touche l'objet du brevet, un tel procédé pourrait être justifié par une exception. Le droit allemand prévoit une exemption légale pour des *expériences* affectant l'objet du brevet[81]. L'avant-projet de l'IFPI pour une révision de la loi sur les brevets d'invention reprend cette disposition[82]. Selon le rapport explicatif[83], des expériences pour des buts de recherche scientifique et pour le développement d'inventions dépendantes[84] sont déjà licites sous l'empire du droit actuel[85]. En fait, le droit des brevets vise à assurer au titulaire le bénéfice économique de son inven-

---

78 Voir sur la nature juridique de l'art. 98 CO WALTER FELLMANN, Die Ersatzvornahme nach Art. 98 Abs. 1 OR – ‹Vollstreckungstheorie› oder ‹Erfüllungstheorie›, recht 1993, p. 109 ss. Si une décompilation au lieu de la prestation exigée remplit les conditions de l'art. 52 al. 3 CO, l'auteur ne peut toutefois pas intenter une action en *dommages-intérêts*.

79 Art. 8 al. 1 LBI. Voir aussi l'art. 64 CBE.

80 En droit des brevets, le Tribunal fédéral maintient la théorie de l'épuisement national tandis que pour le droit de marques et le droit d'auteur, il part de l'épuisement international. Voir pour le droit des brevets ATF 126 III 129 (Kodak), pour le droit d'auteur 124 III 321 (Nintendo) et pour le droit de marques 122 III 469 (Chanel). Puisque l'étendue de l'épuisement a des implications économiques très importantes, cette question reste fortement controversée au niveau suisse et européen. L'avant-projet de l'IFPI pour une révision du LB évite d'ailleurs de prendre position dans ce débat.

81 § 11 ch. 2 PatG RFA. Voir PETER CHROCZIEL, die Benutzung patentierter Erfindungen zu Versuchs- und Forschungszwecken, Cologne/Berlin/Bonn/Munich 1986, p. 170 ss, avec des références à la doctrine allemande et américaine.

82 Voir l'art 10a de l'avant-projet pour une révision de la LBI.

83 Rapport explicatif du 29. 10. 2001, p. 38 ss, ch. 2.1.2.2.3.

84 Voir l'art. 36 LBI

tion. Des intérêts économiques à monopoliser davantage les marchés des produits interopérables, des pièces de rechange ou du développement des produits ne sont pas protégés par l'objet du brevet. Afin de ne pas empêcher des analyses licites selon le droit d'auteur, il convient en tout cas d'interpréter d'une façon large l'exception pour les expériences[86].

IV. Les topographies

L'analyse d'une topographie d'un composant semi-conducteur (puce) est admissible selon la loi sur la protection des topographies de produits semi-conducteurs[87]. Elle permet même la *duplication* de topographies à des fins d'enseignement et de recherche[88] et l'utilisation du savoir-faire acquis par des mesures d'ingénierie inverse pour la création de topographies nouvelles non banales[89].

Si des softwares sont intégrés dans des produits semi-conducteurs, une analyse de la structure corporelle reste donc licite alors que toute sorte de décompilation tombe sous le coup de la loi sur le droit d'auteur.

V. Protection complémentaire par la LCD?

La *relation entre la concurrence et la propriété intellectuelle* a toujours préoccupé la doctrine et la jurisprudence. Le Tribunal fédéral applique cumulativement la loi contre la concurrence déloyale et les lois de la propriété intellectuelle[90]. Il limite toutefois son champ d'application en ce sens qu'une protection refusée par le droit de la propriété intellectuelle ne peut pas être

---

85 Voir aussi PETER HEINRICH, (note 8), n. 8.05 ad art. 8 LBI/art. 64 CBE, et MARIO M. PEDRAZZINI, Zur patentrechtlichen Problematik von Versuchen, die ein fremdes Patentrecht benützen, GRUR Int. 1996, p. 373 ss.

86 Voir aussi l'art. 6 de la proposition de la Commission européenne d'une directive concernant la brevetabilité des inventions mises en œuvre par ordinateur (note 7). Voir pour l'admissibilité de la décompilation d'un logiciel breveté dans le cadre d'expériences portant sur l'objet du brevet en droit allemand MARLY (note 16), p. 298.

87 NICOLAS VON WERDT, Kopierschutz für Computerchips, in: Softwareschutz, éd. par Felix H. Thomann/Georg Rauber, Berne 1998, p. 188 ss, p. 205 s., part de l'idée que l'ingénierie inverse des topographies serait même licite sans la permission de l'art. 7 LTo. Voir pour l'admissibilité de l'analyse des topographies en droit allemand MARLY (note 16), p. 298 ss.

88 Art. 7 al. 1 LTo. Voir aussi NICOLAS VON WERDT, Ausgewählte Probleme zum Topographienschutz von mikro-elektronischen Halbleitererzeugnissen; der Schutzgegenstand und die Schutzvoraussetzungen, thèse Berne 1991, Zurich 1991, p. 181 s.

89 Art. 7 al. 2 LTo et message, FF 1989 III 465 ss, p. 559.

90 Voir ATF 97 II 78 consid. 3, 96 II 243 consid. 7, 96 II 400 consid. 6 et 92 II 257 consid. III/1.

obtenue par le *détour* de la loi contre la concurrence déloyale[91]. Les objectifs de la protection et la perspective de la loi sur le droit d'auteur et de la loi contre la concurrence déloyale sont différents. Le droit de la concurrence ne présuppose aucune atteinte à la propriété intellectuelle, mais un acte déloyal propre à distordre le libre jeu de la concurrence[92].

Certains produits susceptibles d'ingénierie inverse ne sont pas protégés par les lois de la propriété intellectuelle et industrielle. Par exemple, l'ingénierie inverse des *logiciels ordinaires* dépourvus de caractère individuel est licite car le droit d'auteur n'est pas applicable. Même des passages non individuels d'un logiciel partiellement protégé peuvent être librement décompilés[93]. Des limites de décompilation et d'utilisation hors du champ de l'individualité auraient créé une protection de savoir-faire non justifiable par le droit d'auteur. De ce point de vue, il serait peu cohérent d'interdire la libre utilisation de passages non individuels découverts dans le cadre d'une décompilation licite[94].

Selon la jurisprudence du Tribunal fédéral, un comportement conforme aux lois de la propriété intellectuelle ne peut être considéré comme une infraction à la *clause générale* de la loi contre la concurrence déloyale que dans des cas exceptionnels[95]. Contrairement au droit d'auteur, la loi contre la concurrence déloyale ne vise que des comportements avec une influence possible sur la libre-concurrence[96]. Elle ne peut pas sanctionner l'analyse même

---

91 Voir ATF 104 II 322 consid. 5 b et 97 II 78 consid. 3. Cependant, la jurisprudence du Tribunal fédéral a suscité des critiques au sein de la doctrine. Voir STREULI-YOUSSEF, SIWR V/1, 2e éd., p. 155 ss; MÜLLER, SIWR V/1, 2e éd., p. 47 ss.

92 Voir p. ex. TROLLER (note 26), tome II, p. 903 s.

93 Voir aussi HUET, l'Europe des logiciels (note 27), p. 316 s. Toutefois, la *décompilation de logiciels paraissant non individuels* au-delà des conditions de l'art 21 LDA est délicate: sans autres indices d'un manque de protection (par exemple des jugements antérieurs), seul le résultat de la décompilation permet d'apprécier l'admissibilité de la décompilation même. Voir LINANT DE BELLEFONDS (note 38), p. 484.

94 Voir cependant l'art. 6 al. 2 lit. b de la directive 91/250 CE et l'art. 21 al. 2 lit. b de l'avant-projet de l'IFPI pour la révision du droit d'auteur.

95 Voir ATF 112 II 362 consid. 3, 104 II 334 consid. 5 a, et 90 II 56 consid. 6, s'agissant toutefois de marques et de modèles. Jusqu'à présent, l'art. 2 LCD n'a guère été concrétisé par rapport à l'appréhension de secrets.

96 La LCD ne présuppose plus de rapport de concurrence entre le lésant et le lésé, mais les activités doivent avoir un rapport avec le marché pour qu'elles puissent tomber sous son coup. Voir CARL BAUDENBACHER, Lauterkeitsrecht, Kommentar zum Gesetz gegen den unlauteren Wettbewerb (UWG), Bâle/Genève/Munich/St. Gall/Berlin 2001, n. 1 supra art. 2 LCD; BERNARD DUTOIT, Wettbewerbsverhältnis als Erfordernis des Wettbewerbsrechts? Versuch einer Antwort aus schweizerischer und rechtsvergleichender Sicht, SIC 2000, p. 242 ss, et IDEM, Droit de la concurrence déloyale et rapport de concurrence: un coupe indissociable? Un essai de réponse comparative, in: Mélanges en l'honneur de Jacques-Michel Grossen, Bâle/Francfort-sur-le-Main 1992, p. 297 ss, p. 298 ss.

d'un produit, mais l'exploitation d'un secret découvert par des mesures illicites[97].

L'art. 5 lit. c LCD traite plutôt de la *duplication identique de produits entiers* et non pas de l'usage d'informations particulières, découvertes lors de l'ingénierie inverse. Cette disposition sera par contre applicable aux copies pirates, effectuées à l'aide d'une décompilation précédente.

L'art. 6 LCD interdit l'exploitation et la *divulgation* de secrets de fabrication ou d'affaires surpris ou indûment découverts d'une autre manière, mais non pas la découverte elle-même. L'application de cette disposition présuppose d'abord que le produit analysé contienne des secrets, c'est-à-dire des informations de valeur commerciale non accessibles au public[98]. Le code source d'un logiciel peut contenir de telles informations[99]. La loi suggère toutefois que la manière de découvrir doit elle-même être illicite[100]. Par conséquent, l'art. 6 LCD ne contient aucune base suffisante pour interdire la décompilation indépendamment du droit d'auteur.

En conclusion, la loi sur la concurrence déloyale ne limite pas l'ingénierie inverse au-delà du droit de la propriété intellectuelle mais elle peut interdire la divulgation des résultats d'une analyse illicite selon la loi sur le droit d'auteur indépendamment de l'individualité des informations en question.

VI. Conclusions

L'analyse de produits par des mesures d'ingénierie inverse ne met pas en danger la propriété intellectuelle elle-même, mais elle augmente le risque d'infractions et met en péril le savoir-faire non protégé.

La nature technique de toute analyse engendrant une copie d'une œuvre protégée met en jeu le droit de la propriété intellectuelle. Le droit suisse admet la décompilation d'un logiciel en vue de la création de programmes interopérables dans des conditions étroites, analogues à celles du droit communautaire. A part l'art. 21 LDA, l'usage conforme d'un programme d'ordinateur peut à la limite comprendre une décompilation pour autant que les

---

97 Voir Guyet, SIWR V/1, 2e éd., p. 225.

98 Voir Mario M. Pedrazzini, Unlauterer Wettbewerb, Berne 1992, p. 165 ss, et Guyet, SIWR V/1, 2e éd., p. 226. Selon Baudenbacher (note 96), n. 29 ad art. 6 LCD, l'art. 6 LCD ne présuppose par contre aucune obligation de discrétion.

99 Voir aussi la décision du Tribunal cantonal de Nidwald du 15 février 1989 publiée en RSPI 1989, p. 271 s. Voir pour le droit allemand Marly (note 16), p. 302 s.

100 Voir surtout le texte italien (‹affari che ha spiato›) et allemand (‹Auskundschaften›). Selon Pedrazzini (note 98), p. 165 s., surprendre un secret au sens de l'art 6 LCD signifie le repérer en étant conscient qu'il s'agit d'un acte illicite ou d'un cas limite.

informations indispensables à son utilisation ne puissent pas être obtenues par d'autres moyens.

Même si l'ingénierie inverse de produits brevetés engendre d'un point de vue technique des copies, elle doit être admise dans des limites analogues à celles du droit d'auteur. Il convient d'interpréter de façon large une exception pour des expériences.

Tant que l'on reste dans les limites du comportement admis par les lois de la propriété intellectuelle et industrielle, la seule analyse d'un produit ne peut pas porter atteinte à la concurrence déloyale, mais l'exploitation d'informations découvertes par des mesures illicites peut tomber sous le coup de la loi contre la concurrence déloyale.